# 目录　MATLAB 6.5

# 第1章 概　述

在国民经济各部门和科学技术的各个领域中普遍存在着最优化问题，最优化问题就是从所有可能的方案中选择出最合理的、达到最优目标的方案，即最优方案，搜索最优方案的方法就是最优化方法。

本章主要内容：

- MATLAB 6.5 语言简介
- MATLAB 6.5 优化工具箱的特点
- MATLAB 6.5 优化工具箱工程应用简介
- 优化问题的工程背景

## 1.1 引　言

最优化是一个古老的问题，追求最优目标一直是人类的理想，长期以来，人们对最优化问题进行不断的探讨和研究。最优化方法就是从众多可能的解决方案中选择最佳者，以达到最优目标的科学。早在 17 世纪，英国伟大的科学家 Newton 开创了微积分时代，已经提出极值问题；后来出现 Lagrangian 乘数法，1847 年法国数学家 Cancky 研究了函数沿什么方向下降最快的问题；1949 年前苏联数学家 KaHTOPOBNq 提出可解决下料问题和运输问题这两种线性规划问题的求解方法。但是由于受到计算手段等历史条件的限制，在 20 世纪 40 年代以前，最优化理论还不能形成一门学科。

自 20 世纪 40 年代以来，由于生产和科学研究突飞猛进地发展，最优化理论和方法日益受到人们的重视，特别是计算机日益广泛应用，使最优化问题的研究不仅成为一种迫切的需要，而且有了求解的有力工具，因此最优化理论和算法迅速发展起来，形成了一门新的应用数学分支学科，已经渗透到生产、管理、商业、军事、决策等各领域。至今已出现线性规划、整数规划、非线性规划、几何规划、动态规划、随机规划、网络流等许多分支．最优化理论和算法在实际应用中正在发挥着越来越重要的作用。

近几十年来，随着计算机的普遍应用和发展，各种辅助计算或设计软件层出不穷，大大丰富了工程技术人员的开发研究手段，提高了解决问题的计算速度，在工程应用中发挥了重要的辅助作用。本书所说明的 MATLAB 6.5 软件就是其中重要的一种。

自 1994 年美国 MathWorks 公司推出 MATLAB 以来，目前 MATLAB 已发展成为国际上最优秀的科技应用软件之一，它以强大的科学计算与可视化功能、简单易用、开放式可扩展环境，特别是所附带的 30 多种面向不同领域的工具箱支持，使得 MATLAB 在许多科学领域中成为计算机辅助设计和分析、算法研究和应用开发的基本功聚合首选平台。MATLAB 最初用于自动控制系统的辅助设计，而后采用了开放性开发的思想，不断吸收各学科领域所开发的实用程序，形成了一套规模大、覆盖面积广的工具箱，包括信号处理、图像处理、小波分析、系统识别、通信仿真、模糊控制、神经网络、工程优化、统计分析

等许多现代工程技术学科的内容，它的应用范围涵盖了当今所有的工业、电子、医疗、建筑等各领域。今天的工程技术人员面临着如何在短时间内高效出色地完成他们复杂的科研项目的难题，而 MATLAB 的出现使得他们能迅速地测试他们的构想，综合评测系统性能，并能快速设计出更多解决方案来满足未来更多更高的技术要求。

# 1.2 MATLAB 6.5 语言简介

MATLAB 语言是一种非常强大的工程语言，本节主要对该语言进行简单的介绍，包括 MATLAB 的产生背景及主要产品、MATLAB 的语言特点，以及 MATLAB 6.5 新版本所具有的新特点。

## 1.2.1 MATLAB 的产生背景及主要产品

MATLAB 诞生在 20 世纪 70 年代，它的编写者是 Cleve Moler 博士和他的同事。当时，Cleve Moler 博士和他的同事开发了 EISPACK 和 LINPACK 的 FORTRAN 子程序库。这两个程序库主要是求解线性方程的程序库。但是，Cleve Moler 发现学生使用这两个程序库时有困难，主要是接口程序不好写，很费时间。于是 Cleve Moler 自己动手，在业余时间里编写了 EISPACK 和 LINPACK 的接口程序。Cleve Moler 给这个接口程序取名为 MATLAB，意为矩阵（Matrix）和实验室（Laboratory）的组合。以后几年，MATLAB 作为免费软件在大学里使用，深受大学生的喜爱。

1984 年，Cleve Moler 和 John Little 成立了 MathWorks 公司，正式把 MATLAB 推向市场，并继续进行 MATLAB 的开发。1993 年，MathWorks 公司推出 MATLAB 4.0；1995 年，MathWorks 公司推出 MATLAB 4.2C 版（WIN 3.x）；1997 年此公司推出 MATLAB 5.0；2000 年 10 月，MathWorks 公司推出 MATLAB 6.0；2002 年 8 月，新的版本 MATLAB 6.5 已经开始发布了。每一次版本的推出都使 MATLAB 有了长足的进步，界面越来越友好，内容越来越丰富，功能越来越强大。它的帮助信息采用超文本格式和 PDF 格式，可以很方便地阅读。

MATLAB 擅长于数值计算，能处理大量的数据，而且效率比较高。MathWorks 公司在此基础上开拓了符号计算、文字处理、可视化建模和实时控制能力，增强了 MATLAB 的市场竞争力，使 MATLAB 成为市场上主流的数值计算软件。

MATLAB 产品组是支持概念设计、算法开发、建模仿真、实时实现的理想的集成环境。无论是进行科学研究还是产品开发，MATLAB 产品组都是必不可少的工具。

MATLAB 产品组可用来进行：

- 数据分析；
- 数值和符号计算；
- 工程与科学绘图；
- 控制系统设计；
- 数字图像信号处理；

- 财务工程；
- 建模、仿真、原型开发；
- 应用开发；
- 图形用户界面设计。

MATLAB 产品组被广泛地应用于包括信号与图像处理、控制系统设计、通信、系统仿真等诸多领域。开放式的结构使 MATLAB 产品组很容易针对特定的需求进行扩充，从而在不断深化对问题的认识的同时，提高自身的竞争力。

MATLAB 产品组的一大特性是有众多的面向具体应用的工具箱和仿真块，包含完整的函数集，用来对信号图像处理、控制系统设计、神经网络等特殊应用进行分析和设计。其他的产品延伸了 MATLAB 的能力，包括数据采集、报告生成和依靠 MATLAB 语言编程产生独立 C/C++代码等。

MATLAB 主要产品构成如下。

1．MATLAB

所有 MathWorks 公司产品的数值分析和图形基础环境。MATLAB 将 2D 和 3D 图形、MATLAB 语言能力集成到一个单一的、易学易用的环境之中。

2．MATLAB Toolbox

这一系列专用的 MATLAB 函数库可以解决特定领域的问题。工具箱是开放的、可扩展的，可以查看其中的算法，或开发自己的算法。

3．MATLAB Compiler

将 MATLAB 语言编写的 M 文件自动转换成 C 或 C++文件，支持用户进行独立应用开发。结合 MathWorks 提供的 C/C++数学库和图形库，用户可以利用 MATLAB 快速地开发出功能强大的独立应用。

4．Simulink

它是结合了框图界面和交互仿真能力的非线性动态系统仿真工具。它以 MATLAB 的核心数学、图形和语言为基础。

5．Stateflow

与 Simulink 框图模型相结合，描述复杂事件驱动系统的逻辑行为，驱动系统在不同的模式之间进行切换。

6．Real-Time Workshop

直接从 Simulink 框图自动生成 C 或 ADA 代码，用于快速原型和硬件的回路仿真，整个代码的生成可以根据需要完全定制。

7．Simulink Blockset

专门为特定领域设计的 Simulink 功能块的集合，用户也可以利用已有的块或自编写的 C 和 MATLAB 程序建立自己的块。

### 1.2.2 MATLAB 语言的特点

MATLAB 语言有不同于其他高级语言的特点，它被称为第四代计算机语言。正如第三代计算机语言如 FORTRAN 语言与 C 语言等使人们摆脱了对计算机硬件的操作一样，MATLAB 语言使人们从繁琐的程序代码中解放出来。它的丰富的函数使开发者无需重复编程，只需简单地调用和使用。MATLAB 语言最大的特点是简单和直接。MATLAB 语言的主要特点有以下几点。

1. 编程效率高

MATLAB 是一种面向科学与工程计算的高级语言，允许用数学形式的语言编写程序，且比 BASIC、FORTRAN 和 C 等语言更加接近我们书写计算公式的思维方式，用 MATLAB 编写程序犹如在演算纸上排列出公式与求解问题。因此，MATLAB 语言也可以通俗地称为演算纸式科学算法语言。由于它编写简单，所以编程效率高，易学易懂。

2. 用户使用方便

MATLAB 语言是一种解释执行的语言（在没被专门的工具编译之前），它灵活、方便，其调试程序的手段丰富，调试速度快，需要学习的时间少。人们用任何一种语言编写程序和调试程序一般都要经过这样几个步骤：编辑、编译、连接，以及执行和调试。各个步骤之间是顺序关系，编程的过程就是在它们之间做瀑布型的循环。MATLAB 语言与其他语言相比，较好地解决了上述问题，把编辑、编译、连接和执行融为一体。它能在同一画面上进行灵活操作，快速排除输入程序中的书写错误、语法错误及语意错误，从而加快了用户编写、修改和调试程序的速度。可以说，在编程和调试过程中它是一种比 Visual Basic 还要简单的语言。

具体地说，MATLAB 运行时，如直接在命令行输入 MATLAB 语句（命令），包括调用 M 文件的语句，每输入一条语句，就立即对其进行处理，完成编译、连接和运行的全过程。又如，将 MATLAB 源程序编辑为 M 文件，由于 MATLAB 磁盘文件也是 M 文件，所以编辑后的源文件就可直接运行，而不需进行编译和连接。在运行 M 文件时，如果有错，计算机屏幕上会给出详细的出错信息，用户经修改后再执行，直到正确为止。所以，MATLAB 语言不仅是一种语言，广义上讲是一种该语言的开发系统，即语言调试系统。

3. 扩充能力强，交互性好

高版本的 MATLAB 语言有丰富的库函数，在进行复杂的数学运算时可以直接调用，而且 MATLAB 的库函数同用户文件在形成上一样，所以用户文件也可以作为 MATLAB 的库函数来调用。因而，用户可以根据自己的需要方便地建立和扩充新的库函数，以便提高 MATLAB 使用效率和扩充它的功能。另外，为了充分利用 FORTRAN、C 等语言的资源，包括用户已编好的 FORTRAN、C 语言程序，通过建立 M 文件的形式混合编程，方便地调用有关的 FORTRAN、C 语言的子程序；还可以在 C 语言和 FORTRAN 语言中方便地使用 MATLAB 的数值计算功能，这样，良好的交互性使程序员可以使用以前编写过的程序，减少重复性工作，也使现在编写的程序具有重复利用的价值。

4. 可移植性很好，开放性也很好

MATLAB 是用 C 语言编写的，而 C 语言的可移植性很好。于是 MATLAB 可以很方便地移植到能运行 C 语言的操作平台上。MATLAB 适合的工作平台有：Windows 系列、UNIX、Linux、VMS 6.1、PowerMac。除了内部函数外，MATLAB 所有的核心文件和工具箱文件都是公开的，都是可读可写的源文件，用户可以通过对源文件的修改和自己编程构成新的工具箱。

5. 语句简单，内涵丰富

MATLAB 语言中最基本、最重要的成分是函数，其一般形式为［a，b，c，…］= fun（d，e，f，…），即一个函数由函数名、输入变量（如 d，e，f，…）和输出变量（如 a，b，c，…）组成，同一函数名 F，不同数目的输入变量（包括无输入变量）及不同数目的输出变量，代表着不同的含义（有点像面向对象中的多态性）。这不仅使 MATLAB 的库函数功能更丰富，而且大大减少了需要的磁盘空间，使得 MATLAB 编写的 M 文件简单、短小而高效。

6. 高效方便的矩阵和数组运算

MATLAB 语言像 BASIC、FORTRAN 和 C 语言一样规定了矩阵的算术运算符、关系运算符、逻辑运算符、条件运算符及赋值运算符，而且这些运算符大部分可以毫无改变地照搬到数组运算中，如算术运算符只要增加“.”就可用于数组运算。另外，它不需定义数组的维数，并给出矩阵函数、特殊矩阵专门的库函数，使之在求解诸如信号处理、建模、系统识别、控制、优化等领域的问题时，显得大为简捷、高效、方便，这是其他高级语言所不能比拟的。在此基础上，高版本的 MATLAB 已逐步扩展到科学及工程计算的其他领域。因此，不久的将来，它一定能名副其实地成为“万能演算纸式的”科学算法语言。

7. 方便的绘图功能

MATLAB 的绘图是十分方便的，它有一系列绘图函数（命令），例如线性坐标、对数坐标、半对数坐标和极坐标，均只需调用不同的绘图函数（命令）在图上标出图题、*XY* 轴标注。格（栅）绘制也只需调用相应的命令，简单易行。另外，在调用绘图函数时，调整自变量可以绘出不变颜色的点、线、复线或多重线。这种为科学研究着想的设计是通用的编程语言所不及的。

### 1.2.3 MATLAB 6.5 的新特点

在 2002 年，MathWorks 公司发布 MATLAB 6.5 最新版产品。MATLAB 6.5 的特点在于全新的桌面及各种不同领域的集成工具，使用户易于使用。多种新工具简化了一般的工作如资料输入、快速分析，并创造高品质且具有实用性的图表分析等。MATLAB 6.5 包含了新的 JIT 加速度计，JIT 加速度计有力地增加了 MATLAB 中的许多操作和数据类型的计算速度。其他新的特色和加强包括以下三个方面。

1. 编程和数据类型

（1）增加了变量名、函数名和文件名的最大长度，如变量名、函数名、子函数名、

结构域名、M 文件名、MEX 文件名和 MDL 文件名的最大长度可以达到 63 个字节；

（2）支持 64 位的文件偏移量，能够为大于 2GB 的数据文件实现低层次的 I/O 函数；

（3）支持有符号和无符号的 64 位整数；

（4）支持使用动态域名来访问和修改结构数组；

（5）简化了 AND 和 OR 逻辑运算；

（6）支持新的 MATLAB 定时器，而不是定时执行 MATLAB 命令；

（7）改进了音频支持；

（8）加强了警告和错误提示功能，支持格式化的字符串和消息标识符。

**2. 外部接口**

（1）改进了自动化客户接口——新的查看和修改属性用户接口，增强了事件和例外句柄；

（2）增强了网络集成——读 URL 的内容，在 MATLAB 中发送 E-mail 及解压缩文件。

**3. 开发环境**

（1）新的 M-文件接口，能更好地理解 M-代码；

（2）新的启动按钮，易于执行共同的命令；

（3）改进了文件和目录管理工具；

（4）增强了数组编辑器、与 Excel 之间剪切、复制、删除和交换单元的新功能，支持更大的数组；

（5）改进了编辑和调试工具；

（6）改进与 PC 平台的控制接口；

（7）支持新的图形用户接口，从 HDF 或 HDF-EOS 文件导入数据；

（8）JVM 1.3.1 支持 Windows，Linux 和 Solaris 平台。

**4. 图形**

提高了图形性能，如新的彩色图形编辑器，改进了图形特性编辑器。

**5. 数学**

（1）新的数学计算和算法改进；

（2）在 Pentium 4 计算机上更快地计算许多函数，比如矩阵乘法、矩阵转置和线性代数运算。特别指出的是，许多新特色并不适合于所有的平台。

## 1.3 MATLAB 6.5 优化工具箱的特点

MATLAB 6.5 所带的优化工具箱（Optimization Toolbox）是 V2.2 版本的，它在 V2.1 版本的基础上改进了自身性能，除了拥有 V2.1 版本的特点外，V2.2 版本还具有以下几个新特点。

**1. 工具箱的整体运行速度得到了提高**

因为系统提高了函数 optimset 和 optimget 的运行速度，从而使工具箱的整体运行速

度得到了提高（读者如果在工程应用中反复调用工具箱中的函数，就可以发现程序执行需要的时间比以前要短）。

**2. 支持函数句柄**

在实际应用中，你可能需要将一个函数作为另一个函数的参数，这在以前版本中是一件很难办到的事。但在 V2.1 版本中，你可以使用函数句柄这一新的用法方便地实现要求。首先定义一个函数句柄指向一个函数，然后将其传递给另一个函数作为参数。当然也可以像在以前版本中的一样，将函数定义为一个字符串或一个内联对象。



为了了解更多的关于“函数句柄”方面的知识，请读者参考函数 func2str 和 str2func。

**3. 求解复杂结构的优化问题**

在实际工程应用中，你会遇到一类的问题，其中含有大结构的稠密 Hessian 矩阵或者 Jacobian 矩阵，增加了问题求解的难度。但 V2.1 在优化函数的 options 参数中增加了两个新的成员参数：HessMult 和 JacobMult，你可以快速有效地对这两种矩阵优化所需的各种计算，最终方便地解决你的实际问题。

# 1.4 MATLAB 6.5 优化工具箱工程应用简介

上一节介绍了 MATLAB 6.5 优化工具箱的特点，那么它到底有哪些功能呢？怎样使用优化工具箱来解决工程中的实际问题呢？本节将简单讨论优化工具箱的应用。

## 1.4.1 优化工具箱的工程应用功能

MATLAB 6.5 中的优化工具箱（Optimization Toolbox）中含有一系列的优化算法函数，这些函数拓展了 MATLAB 6.5 数字计算环境的处理能力，可以用于解决以下工程实际问题：

- 求解无约束条件非线性极小值；
- 求解约束条件非线性极小值，包括目标逼近问题、极大–极小值问题，以及半无限极小值问题；
- 求解二次规划和线性规划问题；
- 非线性最小二乘逼近和曲线拟合；
- 非线性系统的方程求解；
- 约束条件下的线性最小二乘优化；
- 求解复杂结构的大规模优化问题。



优化工具箱中的所有函数都对应于一个 MATLAB 6.5 的-M 文件（关于-M 文件请查阅相关 MATLAB 6.5 文献），这些-M 文件通过使用 MATLAB 6.5 的基本语句实现了具体的优化算法。你可以在 MATLAB 6.5 命令窗口键入命令：type function_name，来查看相应函数的代码。

### 1.4.2 优化工具箱的工程应用步骤

当量化地求解一个实际的最优化问题时，首先要把这个问题转化为一个数学问题，即建立数学模型；然后对建立的数学模型进行具体分析，选择合适的优化算法；最后根据选定的优化算法，编写计算程序进行求解。用 MATLAB 6.5 优化工具箱解决实际工程应用问题可概括为以下三个步骤：

（1）根据所提出的最优化问题，建立最优化问题的数学模型，确定变量，列出约束条件和目标函数（指标函数和性能函数）；

（2）对所建立的模型进行具体分析和研究，选择合适的最优化求解方法；

（3）根据最优化方法的算法，列出程序框图、选择优化函数和编写语言程序，用计算机求出最优解。



建立数学模型是很重要的一环。数学模型的优劣将关系到问题求解的正确性和合理性。要建立一个合适的数学模型，必须对实际问题有很好的了解，经过分析、研究抓住问题的主要因素、理清其相互的联系，然后综合利用有关学科的知识和数学知识才能完成。

## 1.5 优化问题的工程背景

随着生产、经济、技术的发展，工程技术、管理人才在实际工作中常常会面临这样的一类问题：在工程设计中，怎样选取参数使得设计既满足要求又能降低成本；在资源分配中，怎样的分配方案既能满足各方面的基本要求，又能获得好的经济效益；在生产计划安排中，选择怎样的计划方案才能提高产值和利润；在原料配比问题中，怎样确定各种成分的比例才能提高质量、降低成本；在城建规划中，怎样安排工厂、机关、学校、商店、医院、住宅和其他单位的合理布局，才能方便群众，有利于城市各行各业的发展。这一类问题的共同点就是选出最合理、达到事先预定的最优目标的方案，这就是工程最优化问题。

### 1.5.1 线性规划

在经济生活中，经常遇到在有限的资源（如人力、原材料、资金等）情况下，如何合理安排才能使经济效益达到最大；或者对给定的任务，如何安排统筹现有的资源，完成给定的任务而使花费最小等问题。这类现实中的优化问题都可以用线性规划的数学模型来描述。在运输行业中，经常要将某种物资从一些产地运到其他销售地，而单位物质的运输费用一般来说都与运输距离有关，根据现有的交通网络，解决如何调运从而使总的费用最少。

典型的工程应用问题如图 1-1 所示。

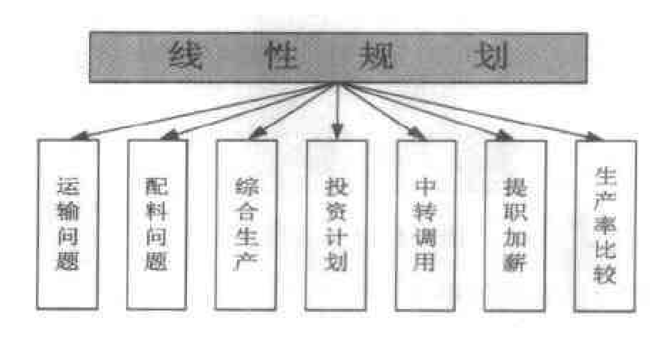



图 1-1 线性规划问题

### 1.5.2 整数规划

整数规划是线性规划模型方法中的一个特例，它的决策变量只限于取非负的整数。例如：销售网点，指派工作人数，最优调度的车辆，下料等。

典型整数规划的工程应用问题如图 1-2 所示。

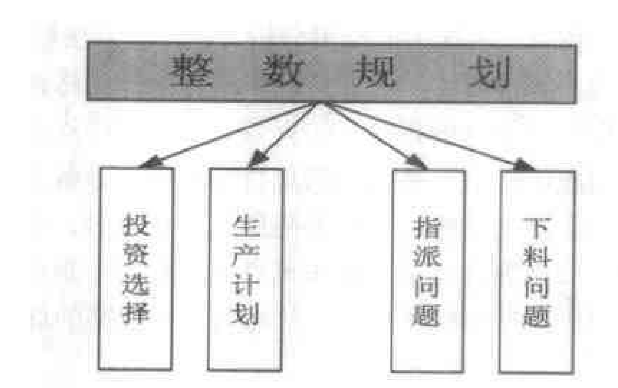



图 1-2 整数规划问题

### 1.5.3 多目标优化决策

在现实活动中，决策的目标往往有许多。例如，对企业产品的生产管理既希望达到高利润，又希望优质和低消耗，还希望减少对环境的污染等，这就是一个多目标决策问题。多目标决策主要有两类，一类是多目标规划问题，其对象是在管理决策过程中求解，使多个目标都达到最满意结果的最优方案；另一类是多目标优选问题，其对象是在管理决策过程中根据多个目标或多个准则衡量后得出各种备选方案的优先等级和排序。

多目标决策模型的应用广泛，其主要应用有：国家发展战略规划、地区发展规划、企业经营管理、工程项目管理、交通运输管理、科研管理、环境保护与管理、工程设计与工艺、公共事业规划、军事国防事业等。

多目标决策问题包含三大要素，如图 1-3 所示。

决策者是提出问题和解决问题并使方案付诸实施的个人或团体，决策者的愿望、需求和偏好影响整个多目标决策问题的形成和解决，决策者的作用是非常重要的。





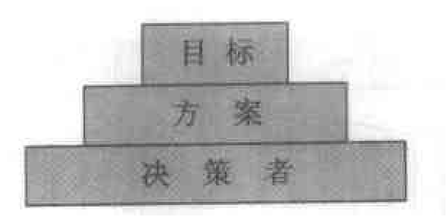




图 1-3　多目标决策问题

解决多目标决策的方法归纳如图 1-4 所示。

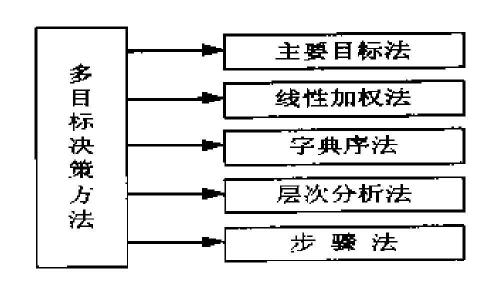



图 1-4　解决多目标决策的方法

层次分析法是一种多目标、多准则的决策分析方法，该方法被广泛应用于工程、经济、军事、政治、外交等领域，解决诸如系统评估、资源分配、价格预测、项目选择等许多重要问题，是一种定性分析和定量分析相结合的有效方法，用层次分析法做决策分析，首先要把问题层次化，根据问题的性质和要达到的总目标将问题分解为不同的组成因素，并按照因素间的相互影响以及隶属关系将因素按不同层次聚集组合，形成一个多层次的分析机构模型，最终把系统分析归结为最低层（如决策方案）相对于最高层（总目标）的相对重要性权值的确定或相对最优次序的排序问题，从而为决策方案的选择提供依据。

### 1.5.4　动态规划

动态规划是一种解决多阶段决策问题的优化方法。其决策过程是一种在多个相互联系的阶段分别做出决策以形成序列决策的过程。而这些决策都是根据总体最优化的目标而采取的。动态规划不仅研究时间变化的决策问题，而且研究非时间因素的决策问题，“阶段”可以是时间意义上的阶段，也可以是空间和一般关系意义上的阶段。动态规划也称多阶段规划，典型问题是最短路问题。

# 第 2 章　优化理论基础及其优化工具箱函数选用

本章将介绍最优化问题的一些基础知识，然后围绕几种常见的最优化问题讨论优化工具箱的使用，主要包括线性规划、无约束非线性规划、约束最优化、多目标规划、大规模优化及最小二乘优化等。为了进一步理解本章所讲述的内容，后面举了一些实例。

本章主要内容包括：

- 最优化问题基础知识
- 线性规划及其优化工具箱函数选用
- 无约束非线性规划及其优化工具箱函数选用
- 约束最优化及其优化工具箱函数选用
- 多目标规划及其优化工具箱函数选用
- 大规模优化问题
- 最小二乘优化及其优化工具箱函数选用
- 工程应用举例

## 2.1　概　　述

20 世纪 30 年代末期，由于军事和工业发展的需要，提出了不能用古典的微分法和变分法解决的最优化问题，在许多学者的努力下，逐渐发展和形成了一些新的数学方法——最优化方法。

### 2.1.1　最优化问题的基本概念

设 $\boldsymbol{x}=(x_1,x_2,x_3,\cdots,x_n)^T$ 为 $n$ 维欧氏空间 $E^n$ 内的一点，$f(\boldsymbol{x})$、$g_i(\boldsymbol{x})\,(i=1,2,\cdots,m)$、$h_i(\boldsymbol{x})\,(i=m+1,\cdots,p)$ 为给定的 $n$ 元函数，则一般的最优化问题的提法是在约束条件：$g_i(\boldsymbol{x})\leqslant 0$，$i=1,2,\cdots,m$ 和 $h_i(\boldsymbol{x})=0$，$i=m+1,\cdots,p$ 之下，求向量 $\boldsymbol{x}$，使函数 $f(\boldsymbol{x})$ 取极小值（或极大值）。这里 $f(\boldsymbol{x})$ 称为目标函数，$g_i(\boldsymbol{x})\leqslant 0$ 称为不等式约束条件，$h_i(\boldsymbol{x})=0$ 称为等式约束条件，$\boldsymbol{x}=(x_1,x_2,x_3,\cdots x_n)^T$ 称为设计变量或决策变量。最优化问题简写为

$$\begin{cases} \min & f(\boldsymbol{x}) \\ s.t. & g_i(\boldsymbol{x}) \leqslant 0, \quad i=1,2,\cdots,m \\ & h_i(\boldsymbol{x})=0, \quad i=m+1,\cdots,p \end{cases} \qquad (式 2.1)$$



求解最优化问题（式 2.1），就是要求（式 2.1）的全局最优解，但在一般情况下，我们往往只能求出它的一个局部最优解。

### 2.1.2 最优化问题分类

**1．线性规划**

若 $f(\boldsymbol{x})$、$g_i(\boldsymbol{x})$、$h_i(\boldsymbol{x})$ 都是 $\boldsymbol{x}$ 的线性函数，则上述问题（式 2.1）称为线性规划问题，简写为 LP 问题。

**2．二次规划**

在（式 2.1）中，若 $g_i(\boldsymbol{x})\ (i=1,2,\cdots,m)$，$h_i(\boldsymbol{x})\ (i=m+1,\cdots,p)$ 都是 $\boldsymbol{x}$ 的线性函数，而 $f(\boldsymbol{x})$ 是 $\boldsymbol{x}$ 的二次函数，则上述问题（式 2.1）称为二次规划问题，常简写为 QP 问题。

**3．非线性规划**

在（式 2.1）中，若函数 $f(\boldsymbol{x})$，$g_i(\boldsymbol{x})\ (i=1,2,\cdots,m)$，$h_i(\boldsymbol{x})\ (i=m+1,\cdots,p)$ 中至少有一个是 $\boldsymbol{x}$ 的非线性函数，则上述问题（式 2.1）称为非线性规划问题，今后常简写为 NLP 问题，显然二次规划是最简单的一种非线性规划。一般的非线性规划问题根据约束条件的情况可分为：

（1）无约束非线性规划问题，即没有任何约束条件限制的非线性规划问题，有时也称之为无约束优化问题。常写成

$$\min \quad f(\boldsymbol{x}), \quad \boldsymbol{x} \in E^n$$

（2）约束非线性规划问题，即在（式 2.1）中至少有一个约束条件限制的非线性规划问题，有时也称之为约束优化问题。

**4．多目标规划**

在（式 2.1）中，若目标函数 $f(\boldsymbol{x})=\left(f_1(\boldsymbol{x}),f_2(\boldsymbol{x}),\cdots,f_p(\boldsymbol{x})\right)^T$，$p \geqslant 2$，即 $f(\boldsymbol{x})$ 是 $\boldsymbol{x}$ 的一个向量函数，则称问题（式 2.1）为多目标规划问题。

### 2.1.3 大规模系统优化问题

科学技术的发展和生产的不断发展使得人们越来越重视大规模系统的优化技术。各种



大型的社会经济系统、通信、运输、分配、能源、企业管理系统的运行处理都是大规模系统的典型例子。由于大规模系统包括多个环节，涉及的因素很多，所以我们不能对各个环节孤立地进行研究和管理，必须把这些环节连接起来，全局地进行研究，以便获得一个全局优化的运行管理系统。这就构成了一个典型的大规模系统优化问题。

由于大规模系统的变量有时达到上百万个，约束多达几十万个，如果以常规的优化算法来解大规模系统，以计算机现在的处理速度，将非常浪费时间，所以人们针对大规模系统提出了一些优化算法，来提高大规模系统优化问题的处理效率。

大规模系统的分类同一般系统一样，可分为大规模系统线性优化和大规模系统非线性优化。



大规模系统优化和一般系统优化的理论基础是一致的，只是大规模系统利用其自身结构的特殊性，发展了新的算法，来减少计算时间。以后我们会在介绍一般系统的优化算法时，对大规模系统的优化算法也向读者略微提及。

# 2.2 线性规划及其优化工具箱函数选用

线性规划是数学规划的一个重要分支，也是最简单、最基础的一类问题，它的历史比较悠久，理论比较成熟，方法也较为完善，用于线性规划的 MATLAB 函数主要是 linprog。本节还要举几个线性规划的例子。

## 2.2.1 基本理论

规划问题的数学模型包含三个组成要素：（1）决策变量，指决策者为实现规划目标采取的方案、措施，是问题中要确定的未知量；（2）目标函数，指问题要达到的目的要求，表示为决策变量的函数；（3）约束条件，指决策变量取值时受到的各种可用资源的限制，表示为含决策变量的等式或不等式。如果在规划问题的数学模型中，决策变量为可控的连续变量，目标函数和约束条件都是线性的，这类模型称为线性规划问题的数学模型。

### 2.2.1.1 线性规划的标准形式

我们规定线性规划的标准型为：

$$\min_{s.t.} \quad c_1x_1 + c_2x_2 + \cdots + c_nx_n$$

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \qquad\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m \\ x_1 \geqslant 0, \quad x_2 \geqslant 0, \quad \cdots, \quad x_n \geqslant 0 \end{cases} \qquad \text{（式 2.2）}$$

其中 $a_{ij}$、$b_i$、$c_i$ 均为实常数，且 $b_i \geqslant 0$，$(i=1,2,\cdots, m)$

上述线性规划问题可简写为

$$\begin{cases} \quad \min \quad \mathbf{c}^T \boldsymbol{x} \\ s.t. \quad \boldsymbol{Ax}=\boldsymbol{b}, \quad \boldsymbol{x} \geqslant 0 \end{cases} \qquad \text{（式 2.3）}$$

其中：$\boldsymbol{A}=(a_{ij})_{m\times n}$，$\boldsymbol{x}=(x_1, x_2,\cdots, x_n)^T$，$\text{rank}(\boldsymbol{A})=m \leqslant n$，

$$\boldsymbol{c}=(c_1, c_2,\cdots, c_n)^T, \boldsymbol{b}=(b_1, b_2,\cdots, b_m)^T, \text{且} \boldsymbol{b} \geqslant 0.$$

实际问题中的线性规划的模型是多种多样的，但可以使用一些方法将它们化为等价的标准型。下面给出一些优化标准型的方法。

**1．极大问题的变换**

假如我们要求目标函数实现最大值，即

$$\max \quad z = c^T \boldsymbol{x}$$

只要将目标函数的最大值化为求目标函数最小值即可

$$\max \quad z = \min \quad (-z)$$

这就和标准型一致了。

**2．不等式约束的变换**

可以通过增加松弛变量使不等式

$$\sum_{j=1}^{n} a_{ij} x_j \leqslant b_i$$

成为等式约束

$$\sum_{j=1}^{n} a_{ij} x_j + x_{n+i} = b_i, \quad x_{n+i} \geqslant 0$$

称 $x_{n+i}$ 为松弛变量。对于反向不等式情况，可引进剩余变量 $x_{n+i}$，使不等式

$$\sum_{j=1}^{n} a_{ij} x_j \geqslant b_i$$

变为等式约束

$$\sum_{j=1}^{n} a_{ij} x_j - x_{n+i} = b_i, \quad x_{n+i} \geqslant 0$$

3. 自由变量

如果某些变量是无约束的（即不一定为非负的），则可通过附加变量的办法把这些变量改写为标准非负约束的形式，事实上，具有任意符号的 $x_k$ 能变成

$$x_k = x_k' - x_k''$$

其中 $x_k' \geqslant 0$，$x_k'' \geqslant 0$。

#### 2.2.1.2 线性规划的基本原理

对于标准形式的线性规划（式 2.2）的可行解等价于线性方程组 $\boldsymbol{Ax}=\boldsymbol{b}$ 的非负解，因线性方程组 $\boldsymbol{Ax}=\boldsymbol{b}$ 有 $n$ 个变量和 $\text{rank}(\boldsymbol{A})=m$，因此当 $\boldsymbol{Ax}=\boldsymbol{b}$ 的一个解中非负分量所对应 $\boldsymbol{A}$ 的列为线性无关解时，则非零分量的个数就不会超过 $m$，也即零分量的个数不少于 $n-m$。称 $\boldsymbol{A}$ 中对应于非零分量的列呈线性无关的 $\boldsymbol{Ax}=\boldsymbol{b}$ 的解为线性规划（式 2.2）的基本解。

可以证明，对于线性规划（式 2.2），有：

（1）若存在一个可行解，则必存在一个基本可行解。

（2）若存在一个最优可行解，则必存在一个最优基本可行解。

#### 2.2.1.3 单纯形法

单纯形法是求解线性规划的一种有效的算法，但是它不仅是求解线性规划的基本方法，而且是非线性规划某些算法的基础。本节介绍单纯形法的基本原理和计算方法。

求解线性规划可采取如下步骤：求得一个基本可行解；检查该基本可行解是否为最优解；若不是，则设法再求一个没检查过的基本可行解，如此继续，直至检查出某基本可行解为最优解为止。这就是所谓的单纯形法。

### 2.2.2 优化工具箱函数选用

在 MATLAB 6.5 优化工具箱中，用于求解线性规划的函数有 linprog，用法如下：

【语法】

```
x = linprog (f, A, b, Aeq, beq)
x = linprog (f, A, b, Aeq, beq, lb, ub)
x = linprog (f, A, b, Aeq, beq, lb, ub, x0)
x = linprog (f, A, b, Aeq, beq, lb, ub, x0, options)
[ x, fval ] = linprog (...)
[ x, fval , exitflag] = linprog (...)
[ x, fval , exitflag, output] = linprog (...)
[ x, fval , exitflag, output, lambda] = linprog (...)
```

【说明】

f：是优化参数 $x$ 的系数矩阵；

lb，ub：设置优化参数 $x$ 的上下界；
fval：返回目标函数在最优解 $x$ 点的函数值；
exitflag：返回算法的终止标志；
output：返回优化算法信息的一个数据结构.

### 2.2.3 工程应用举例

**【工程应用背景】**

已知约束条件（式 2.4），求解 $\min\limits_{x} \boldsymbol{f}^T\boldsymbol{x}$ 。其中 ***f, x, b, beq, lb*** 和 ***ub*** 均是向量；***A*** 和 ***Aeq*** 是矩阵。

$$\begin{cases} \boldsymbol{A} * \boldsymbol{x} \leqslant \boldsymbol{b} \\ \boldsymbol{Aeq} * \boldsymbol{x} = \boldsymbol{beq} \\ \boldsymbol{lb} \leqslant \boldsymbol{x} \leqslant \boldsymbol{ub} \end{cases} \qquad \text{（式 2.4）}$$

**【实例分析】**

求使函数 $f(x) = -5x_1 - 4x_2 - 6x_3$ 取最小值的 $x$ 值，且满足约束条件：

$$\begin{cases} x_1 - x_2 + x_3 \leqslant 20 \\ 3x_1 + 2x_2 + 4x_3 \leqslant 42 \\ 3x_1 + 2x_2 \leqslant 30 \\ x_1 \geqslant 0, x_2 \geqslant 0, x_3 \geqslant 0 \end{cases}$$

例程 2-1 是求解的 MATLAB 代码。

例程 2-1

```
f = [-5; -4; -6];
A = [1, -1, 1; 3, 2, 4; 3, 2, 0];
b = [20; 42; 30];
lb = zeros(3, 1);
[x, fval ] = linprog (f, A, b, [ ], [ ], lb);
```

**【结果输出】**

```
x =
     0.0000
     15.0000
     3.0000
fval =
     -78.0000
```



# 2.3　无约束非线性规划

线性规划的目标函数和约束条件都是其自变量的线性函数，如果目标函数或约束条件中包含自变量的非线性函数，则这样的规划问题就属于非线性规划。有些实际问题可以表达成线性规划问题，但有些实际问题则需用非线性规划的模型来表达，借助于非线性规划的解法来求解。有约束问题与无约束问题是非线性规划的两大类问题，它们在处理方法上有明显的不同。本节主要针对无约束问题，讨论它的 MATLAB 解决方法。

## 2.3.1　基本理论

这部分主要介绍无约束问题的基本原理和基本方法。由于无约束最优化方法在许多工业和科学问题上有很多的应用，并且是有约束问题方法的基础，近几十年来，吸引了许多专家对它进行研究，发展非常迅速。这一部分将精选其中一部分可靠性强、有效性高，并且富有生命力的方法，以提供一个关于无约束最优化方法发展现状的梗概介绍。另一方面，这些内容对于后面有约束最优化方法的研究，也是一个必不可少的知识阶梯。

顺便指出，近年来的有约束方法大大减少了对于无约束方法的依赖程度。但是，在某种意义上却更进一步表现出两者之间在算法构造上的一致性，以及在发展过程中所出现的相互影响和相互渗透。

### 2.3.1.1　最优性条件

局部极小点的一阶必要条件：设函数 $f\left(\boldsymbol{x}\right)$ 在点 $\boldsymbol{x}^{(0)}$ 处可微，且 $\boldsymbol{x}^{(0)}$ 为局部极小点，则必有梯度 $\nabla f\left(\boldsymbol{x}^{(0)}\right)=0$。

在一元函数的讨论中，已知满足一阶必要条件的点未必是局部极小点，对于多元函数当然也是如此，因此可仿照一元函数的情形，利用泰勒公式来导出局部极小点的充分条件。

利用局部极小点的一阶必要条件，求函数极值的问题往往化为求解 $\nabla f\left(\boldsymbol{x}\right)=0$，即求 $\boldsymbol{x}$ 使满足

$$\begin{cases}\dfrac{\partial f\left(\boldsymbol{x}\right)}{\partial\left(x_1\right)}=0\\ \quad\vdots\\ \dfrac{\partial f\left(\boldsymbol{x}\right)}{\partial x_n}=0\end{cases} \tag{式 2.5}$$

问题。这是含有 $n$ 个未知量、$n$ 个方程的方程组，并且一般是非线性的。对于一些特殊情形，如目标函数是二次函数，因而方程组（式 2.5）是线性方程组时，有时可以解出。一

般来说，非线性方程组的求解与求无约束极值一样也是一个难题，甚至前者比后者更困难，一般很难用解析方法求解，这时必须用数值方法求出非线性方程的解。但是，与其用数值计算方法求解（式 2.5），倒不如用数值方法直接求极值。所以下面介绍求解无约束极值问题的常用数值解法。

数值解法常用的是迭代法。迭代法的基本思想是：在给出极小点位置的一个初始估计 $\boldsymbol{x}^{(0)}$ 后，计算一系列的 $\boldsymbol{x}^{(k)}(k=1,2,\cdots)$，希望点列 $\boldsymbol{x}^{(k)}$ 的极限 $\boldsymbol{x}^*$ 就是 $f(\boldsymbol{x})$ 的极小点。下面的问题是怎样产生这样一个点列。当有了迭代点 $\boldsymbol{x}^{(k)}$，下一个迭代点为 $\boldsymbol{x}^{(k+1)}$，二者之差是一个向量，就是 $\boldsymbol{x}^{(k)}$ 指向 $\boldsymbol{x}^{(k+1)}$ 的向量，而一个向量总是由方向和长度所决定的，即总可以写为

$$\boldsymbol{x}^{(k+1)}-\boldsymbol{x}^{(k)}=\lambda_k\boldsymbol{d}^{(k)}$$

或

$$\boldsymbol{x}^{(k+1)}=\boldsymbol{x}^{(k)}+\lambda_k\boldsymbol{d}^{(k)}$$

其中 $\boldsymbol{d}^{(k)}$ 为一个向量，$\lambda_k$ 为步长。当 $\boldsymbol{d}^{(k)}$ 和 $\lambda_k$ 确定后，由 $\boldsymbol{x}^{(k)}$ 就可以惟一地确定 $\boldsymbol{x}^{(k+1)}$。于是依次下去，从给定的 $\boldsymbol{x}^{(0)}$ 便可以求出 $\boldsymbol{x}^{(1)}$，由 $\boldsymbol{x}^{(1)}$ 确定 $\boldsymbol{x}^{(2)}$ 等，从而得到一个点列 $\left\{\boldsymbol{x}^{(k)}\right\}$。如果这个点列逼近我们要求的极小点，我们便称这个序列为极小化序列。所以对于每一个迭代点 $\boldsymbol{x}^{(k)}$，如能设法给出 $\boldsymbol{d}^{(k)}$ 和 $\lambda_k$，则算法也就决定了。各种迭代方法的区别在于得出方向 $\boldsymbol{d}^{(k)}$ 和步长 $\lambda_k$ 的方式不同，特别是方向的产生在方法中起着关键的作用。

既然选取的方法有多种多样，那么应该怎样建立它们的原则呢？很显然，我们要求构造的序列对应的函数值应该是逐渐减小的，即具有

$$f\left(\boldsymbol{x}^{(0)}\right)\geqslant f\left(\boldsymbol{x}^{(1)}\right)\geqslant\cdots\geqslant f\left(\boldsymbol{x}^{(k)}\right)\geqslant\cdots$$

这种性质的算法称为下降算法。

另一个要求是：算法应该具有收敛性，即所产生的序列具有这样的性质，或者序列中的某一点 $\boldsymbol{x}^{(N)}$ 本身是 $f(\boldsymbol{x})$ 的极小点；或者有一个极限点 $\boldsymbol{x}^*$，它是函数 $f(\boldsymbol{x})$ 的极小点。这个要求应该是必不可少的。因为极小化序列不能收敛到极小点，那么我们构造的序列与函数的极小点没有关系，因而也就失去意义了。但是这个要求却不容易做到。

一般把最优化算法的迭代过程分为如下的四步：

（1）选择初始点 $x^{(0)}$，各种方法、各类函数对初始点的要求不尽相同，但总是越靠近最优解越好。

（2）如果得出的迭代点 $x^{(k)}$ 不是最优解，我们建立一套规律以产生方向 $d_k$，使目标函数 $f(x)$ 从 $x^{(k)}$ 出发，沿方向 $d_k$ 可以找到 $x^{(k+1)}$，使得 $f(x)$ 有所下降。

（3）方向 $d_k$ 确定后，在射线 $x^{(k)}+\lambda d^{(k)}(\lambda \geq 0)$ 上选取步长 $\lambda_k$，使 $f\left(x^{(k)}+\lambda d^{(k)}\right)<f\left(x^{(k)}\right)$。如此确定出下一个点 $x^{(k+1)}=x^{(k)}+\lambda_k d^{(k)}$。在大多数的算法中，$\lambda_k$ 的选取使 $f(x)$ 下降的最多，即沿射线 $\mathbf{x}^{(k)}+\lambda\mathbf{d}^{(k)}$ 求 $f(x)$ 的极小值。这是单变量 $\lambda$ 的函数求极小点的问题，称为一维搜索，也称为线性搜索。

（4）检验新的迭代点 $x^{(k+1)}$ 是否为最优解，或者是否为最优解的近似解，检验的方法可因算法和函数的不同而不同。例如，梯度的模小于预选指定的正数 $\varepsilon$，即

$$\left\|\nabla f\left(x^{(k+1)}\right)\right\| \leqslant \varepsilon$$

我们就认为 $x^{(k+1)}$ 是近似最优解，迭代过程也就终止了，否则继续迭代。

#### 2.3.1.2　一维搜索

一维搜索可归结为单变量函数的极小化问题。设目标函数为 $f(x)$，过点 $x^{(k)}$ 沿方向 $d^{(k)}$ 的直线可用点集表示，记为

$$L=\left\{x \mid x=x^{(k)}+\lambda d^{(k)},-\infty<\lambda<+\infty\right\}$$

求 $f(x)$ 在直线 L 上的极小点转化为求一元函数

$$\varphi(\lambda)=f\left(x^{(k)}+\lambda d^{(k)}\right)$$

的极小点。

如果 $\varphi(\lambda)$ 的极小点为 $\lambda_k$，通常称 $\lambda_k$ 为沿方向 $d^{(k)}$ 的步长因子，或简称步长，那么函数 $f(x)$ 在直线上的极小点就是

$$x^{(k+1)}=x^{(k)}+\lambda_k d^{(k)}$$

一维搜索方法除利用高等数学中所讲的方法，即解方程 $\frac{\mathrm{d}\varphi(\lambda)}{\mathrm{d}\lambda}=0$ 外，还有两类方法：一类是试探法，通过一系列点的比较确定极小点；一类是函数逼近法，即用简单的曲线近似代替原来的曲线，用近似曲线的极小点来估计原来曲线的极小点。下面介绍较为常用的方法。

1．牛顿法

牛顿法的基本思想是：在迭代点附近用二阶泰勒公式多项式来近似目标函数 $f(x)$，进而求出极小点的估计值。

考虑问题

$$\min_{x\in R^1} f(x) \qquad \text{（式 2.6）}$$

令 $\varphi(x)=f\left(x^{(k)}\right)+f'\left(x^{(k)}\right)\left(x-x^{(k)}\right)+\frac{1}{2}f''\left(x^{(k)}\right)\left(x-x^{(k)}\right)^2$，再令 $\varphi'(x)=f'\left(x^{(k)}\right)+f''\left(x^{(k)}\right)\left(x-x^{(k)}\right)=0$

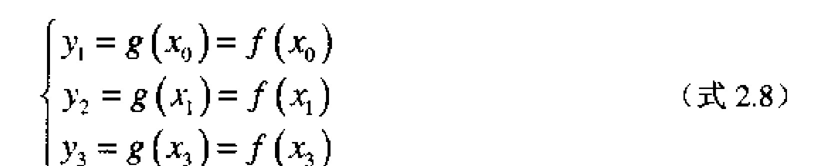
得

$$x^{(k+1)}=x^{(k)}-\frac{f'\left(x^{(k)}\right)}{f''\left(x^{(k)}\right)} \qquad \text{（式 2.7）}$$

用迭代公式（式 2.7）可得到一个点列 $\left\{x^{(k)}\right\}$，在一定条件下，这个点列收敛于（式 2.6）的最优解，且具有二阶收敛速度。

2．抛物线法

抛物线法的基本思想也是利用二次函数 $g(x)$ 来近似代替原来的函数 $f(x)$，并以它的极小点作为函数 $f(x)$ 的近似极小点。它与牛顿法不同的地方在于：牛顿法是用 $f(x)$ 在 $x_0$ 点的二阶泰勒展开式来逼近的，即利用函数值 $f(x_0)$ 及其一、二阶导数值 $f'(x_0)$、$f''(x_0)$ 来构造二次函数；而抛物线法是利用三个点的函数值来构造函数的，即做一个二次函数 $g(x)$，使它满足




$$\begin{cases} y_1 = g(x_0) = f(x_0) \\ y_2 = g(x_1) = f(x_1) \\ y_3 = g(x_3) = f(x_3) \end{cases} \qquad \text{（式 2.8）}$$

其中

$$g(x) = a_0 + a_1 x + a_2 x^2$$

则$g'(x) = a_1 + 2a_2 x$，令$g'(x) = 0$，得

$$x^* = -a_1 / 2a_2 \qquad \text{（式 2.9）}$$

（式 2.9）就是计算$f(x)$的近似极小点的公式。经推导，可得

$$\begin{cases} \left(f(x_1) - f(x_0)\right)/(x_1 - x_0) = b_1 \\ \left(f(x_2) - f(x_1)\right)/(x_2 - x_1) = b_2 \\ a_2 = (b_2 - b_1)/(x_2 - x_1) \\ a_1 = b_1 - a_2(x_1 + x_0) \end{cases}$$

由此可求得$f(x)$的近似极小点。

**3．三次插值法**

三次插值法用$a$、$b$两点处的函数值$f(a)$、$f(b)$和导数$f'(a)$、$f'(b)$来构造三次插值多项式$g(x)$，并以它的极小点作为函数$f(x)$的近似极小点。一般说来，三次插值法比抛物线法收敛速度要快。

设插值多项式为

$$g(x) = \alpha(x-a)^3 + \beta(x-a)^2 + \gamma(x-a) + \delta \qquad \text{（式 2.10）}$$

令$g'(x) = 0$，求$f(x)$的近似极小点。把$f(a)$、$f(b)$、$f'(a)$、$f'(b)$代入（式 2.10），通过推导可得到

$$x^* = \alpha - \frac{\gamma}{\beta + \sqrt{\beta^2 - 3\alpha\gamma}}$$

其中

$$\delta = f(a)$$

$$\gamma = f'(a)$$

$$u = \frac{f(b) - \delta}{b - a} - \gamma$$

$$v = f'(b) - \gamma$$

$$\beta = \frac{3u - v}{b - a}$$

$$\alpha = \frac{v - 2u}{(b-a)^2}$$

#### 2.3.1.3 最速下降法和共轭梯度法

上一小节介绍了一维搜索的方法，本小节和下一小节将介绍确定搜索方向的方法。

**1. 最速下降法**

考虑到函数 $f(\boldsymbol{x})$ 在点 $\boldsymbol{x}^{(k)}$ 处沿着方向 $\boldsymbol{d}$ 的导数 $f_d\left(\boldsymbol{x}^{(k)}\right) = \nabla f\left(\boldsymbol{x}^{(k)}\right)^T \boldsymbol{d}$，其意义为：$f(\boldsymbol{x})$ 在点 $\boldsymbol{x}^{(k)}$ 处沿方向 $\boldsymbol{d}$ 的变化率。当 $f(\boldsymbol{x})$ 连续可微时，方向导数为负，说明函数值沿着该方向下降；方向导数越负，表明下降得越快。因此，确定搜索方向 $\boldsymbol{d}^{(k)}$ 的一个方法就是以 $f(\boldsymbol{x})$ 在点 $\boldsymbol{x}^{(k)}$ 处方向导数最小的方向作为搜索方向，即令

$$\boldsymbol{d}^{(k)} = -\nabla f\left(\boldsymbol{x}^{(k)}\right)$$

因此，也把此方法定名为最速下降法。

求解问题

$$\min_{x \in R^n} f(\boldsymbol{x}), f \in C^1$$

最速下降法的具体步骤为：

（1）选定初始点 $\boldsymbol{x}^{(1)}$ 并给定精度要求 $\boldsymbol{\varepsilon} > 0$，令 $k = 1$；

（2）若 $\left\|\nabla f\left(\boldsymbol{x}^{(k+1)}\right)\right\| \leqslant \boldsymbol{\varepsilon}$，则停止，$\boldsymbol{x}^* = \boldsymbol{x}^{(k)}$；否则 $\boldsymbol{d}^{(k)} = -\nabla f\left(\boldsymbol{x}^{(k)}\right)$；

（3）在 $\boldsymbol{x}^{(k)}$ 处沿方向做线性搜索得 $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{d}^{(k)}$，$k = k+1$，返回（2）。

若在（3）中，不用第一节介绍的近似方法做线性搜索，而是用精确搜索，即

$$\alpha_k = \arg\min f\left(\boldsymbol{x}^{(k)} + \alpha \boldsymbol{d}^{(k)}\right)$$

于是，就有

$$\left.\frac{\mathrm{d}f\left(\boldsymbol{x}^{(k)}+\alpha\boldsymbol{d}^{(k)}\right)}{\mathrm{d}\alpha}\right|_{\alpha=\alpha_k}=\boldsymbol{d}^{(k)T}\nabla f\left(\boldsymbol{x}^{(k+1)}\right)=0 \qquad (式 2.11)$$

（式 2.11）表明$\boldsymbol{d}^{(k)}$和方向$\boldsymbol{d}^{(k+1)}$是正交的。

可以证明，最速下降法是收敛的，最速下降法的计算量不大，它的缺点是收敛速度慢，特别是迭代点接近最优点时，每次迭代行进的距离越来越短。

**2．共轭梯度法**

共轭梯度法是针对二次函数

$$f(\boldsymbol{x})=\frac{1}{2}\boldsymbol{x}^T\mathrm{Q}\boldsymbol{x}+\boldsymbol{b}^T\boldsymbol{x}+c,\ \boldsymbol{x}=(x_1,\ x_2,\cdots,\ x_n)^T \qquad (式 2.12)$$

的无约束极小问题，考虑出一种搜索方向的合理选取方法，然后推广到一般的可微函数。

首先注意到，对于变量分离的函数

$$f(\boldsymbol{x})=f_1(x_1)+f_2(x_2)+\cdots+f_n(x_n)$$

则从任意一点$\boldsymbol{x}^{(1)}$出发，分别沿每个坐标轴方向进行一维搜索，进行一遍（共进行 $n$ 次线性搜索）以后，一定就能得到$\min f(\boldsymbol{x})$的最优解。

而对于形式为（式 2.12）的二次函数，其中$\boldsymbol{Q}$为实对称正定矩阵，只要我们适当选取$\boldsymbol{R}^n$的一组基$\{\boldsymbol{p}_1,\boldsymbol{p}_2,\cdots,\boldsymbol{p}_n\}$，使得$\boldsymbol{p}_i$满足条件

$$\boldsymbol{p}_i^T\boldsymbol{Q}\boldsymbol{p}_j=0 \quad (i\neq j)$$

则显然在新的基下，$f(\boldsymbol{x})$就成为变量分离的形式。于是，从任何一个初始点$\boldsymbol{x}^{(1)}$出发，分别沿每个$\boldsymbol{p}_i$方向做线性搜索，经过一轮后，肯定就能得到最优解。我们把满足条件（式 2.12）的 $n$ 维方向称为是$\boldsymbol{Q}$-共轭的。

基于上面的考虑，现在的问题是如何构造出两两$\boldsymbol{Q}$-共轭的方向？

共轭梯度法就是在每个迭代点$\boldsymbol{x}^{(k)}$处，以负梯度$-\nabla f\left(\boldsymbol{x}^{(k)}\right)$和前一个搜索方向$\boldsymbol{p}_{k-1}$的适当组合，构成和前面的搜索方向$\boldsymbol{p}_1,\boldsymbol{p}_2,\cdots,\boldsymbol{p}_{k-1}$均两两$\boldsymbol{Q}$-共轭的搜索方向$\boldsymbol{p}_k$。（具体的推导参见有关书籍）

求解 $\min f(\boldsymbol{x})=\frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x}+\boldsymbol{b}^T\boldsymbol{x}+c$ 的共轭梯度法的步骤为：

（1）任选初始点 $\boldsymbol{x}^{(1)}\in\boldsymbol{R}^n$，令 $\boldsymbol{p}_1=-\nabla f\left(\boldsymbol{x}^{(1)}\right)$，$k=1$。

（2）若 $\nabla f\left(x^{(k)}\right)=0$，则停；否则

$$\boldsymbol{x}^{(k+1)}=\boldsymbol{x}^{(k)}+\alpha_k\boldsymbol{p}_k,\alpha_k=\frac{-\boldsymbol{p}_k\nabla f\left(\boldsymbol{x}^{(k)}\right)}{\boldsymbol{p}_k^T\boldsymbol{Q}\boldsymbol{p}_k}$$

$$\boldsymbol{p}_{k+1}=-\nabla f\left(\boldsymbol{x}^{(k+1)}\right)+\lambda_k\boldsymbol{p}_k,\lambda_k=\frac{\boldsymbol{p}_k^T\boldsymbol{Q}\nabla f\left(\boldsymbol{x}^{(k+1)}\right)}{\boldsymbol{p}_k^T\boldsymbol{Q}\boldsymbol{p}_k}$$

（3）$k=k+1$，回（2）。

#### 2.3.1.4 牛顿法和拟牛顿法（变尺度法）

1．牛顿法

考虑

$$\min f(\boldsymbol{x})=\frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x}+\boldsymbol{b}^T\boldsymbol{x}+c$$

由于已知

$$\nabla f(\boldsymbol{x})=\boldsymbol{Q}\boldsymbol{x}+\boldsymbol{b}$$

那么由最优性条件 $\nabla f(\boldsymbol{x})=0$，当 $\boldsymbol{Q}$ 正定时，$\boldsymbol{Q}^{-1}$ 存在，立即可得

$$\boldsymbol{x}^*=-\boldsymbol{Q}^{-1}\boldsymbol{b} \qquad \text{（式 2.13）}$$

而且由 $\boldsymbol{Q}$ 的正定性，知 $\boldsymbol{x}^*$ 即为最优解。

于是对于 $f\in C^2$ 的一般函数，在 $\boldsymbol{x}^{(k)}$ 的局部

$$\begin{aligned}f(\boldsymbol{x})\approx{}& f\left(\boldsymbol{x}^{(k)}\right)+\nabla f\left(\boldsymbol{x}^{(k)}\right)^T\left(\boldsymbol{x}-\boldsymbol{x}^{(k)}\right)\\&+\frac{1}{2}\left(\boldsymbol{x}-\boldsymbol{x}^{(k)}\right)^T\nabla^2 f\left(\boldsymbol{x}^{(k)}\right)\left(\boldsymbol{x}-\boldsymbol{x}^{(k)}\right)\end{aligned}$$

当 $\nabla^2 f\left(\boldsymbol{x}^{(k)}\right)$ 正定时，套用（式 2.13），这就是牛顿法，其步骤为：

（1）任取 $\boldsymbol{x}^{(1)}\in\boldsymbol{R}^n$，$k=1$；

（2）计算梯度向量 $\boldsymbol{g}_k=\nabla f\left(\boldsymbol{x}^{(k)}\right)$，若 $\boldsymbol{g}_k=0$，则停止；否则计算 Hessian 矩阵

$G\left(x^{(k)}\right)=\nabla^2 f\left(x^{(k)}\right)$，令

$$x^{(k+1)}=x^{(k)}-G\left(x^{(k)}\right)^{-1}g_k$$

（3）$k=k+1$，回（2）。

**说明** 若 $f(x)$ 为一元函数，则牛顿法实际上就是用切线法解方程 $f'(x)=0$。牛顿法收敛时，收敛速度是很快的（至少是二阶收敛速率），但若初始点选择不好，则有可能不收敛。

在实际应用牛顿法时，如何选取初始点是一个难以解决的问题，于是想到把牛顿法修改为具有整体收敛性（即不依赖初始点的选取）的下降算法。注意到当 $\nabla f(x)\neq 0$ 且 $\nabla^2 f(x)$ 正定时，牛顿方向 $-G(x)^{-1}g$ 是一个下降方向。事实上，由 $\nabla^2 f(x)$ 的正定性有

$$\nabla f(x)^T\left(-\nabla^2 f(x)\right)\nabla f(x)<0$$

于是牛顿法中指定迭代为

$$x^{(k+1)}=x^{(k)}-G\left(x^{(k)}\right)^{-1}g_k$$

修改为沿搜索方向为 $-G\left(x^{(k)}\right)^{-1}g_k=d^{(k)}$ 做线性搜索的迭代点，即：

$$x^{(k+1)}=x^{(k)}+\alpha_k d^{(k)},$$
$$a_k=\arg\min f\left(x^{(k)}+\alpha d^{(k)}\right)$$

2．拟牛顿法

拟牛顿法具有全局收敛性和收敛快的特点，但还有一个缺点，即在每一步确定搜索方向 $d^{(k)}=-G\left(x^{(k)}\right)^{-1}g_k$ 时，要计算 Hessian 矩阵及其逆矩阵 $G\left(x^{(k)}\right)^{-1}$，这样的计算工作量很大。

于是将确定搜索方向 $d^{(k)}$ 公式中的 $G\left(x^{(k)}\right)^{-1}$ 用 $n$ 阶矩阵 $H_k$ 代替。从而在第 $k$ 步迭代时，有：

$$d^{(k)}=-H_k\nabla f\left(x^{(k)}\right),\ \ x^{(k+1)}=x^{(k)}+\alpha_k d^{(k)}$$

$\alpha_k$ 由线性搜索得到，其中初始点 $x^{(1)}$ 的初始矩阵 $H_1$ 是预先给定的，$H_k$ 在迭代中利用已得到的迭代点及目标函数值，作为 $G\left(x^{(k)}\right)^{-1}$ 的近似来构造。

拟牛顿法是一类算法，其中最常用的算法有 DFP 算法和 BFGS 算法。

3．离散牛顿法

上面讨论的都是利用函数导数的方法。有时，很难得到函数的一阶和二阶导数表达式，或此计算代价很大。这时，可考虑采用有限差分技术来近似 Hessian 阵的离散牛顿法，它基本上等价于牛顿法。

对于一个充分小的$\boldsymbol{\delta}_j$，梯度向量的泰勒展开为

$$g(x+\delta_j e_j)\approx g(x)+\delta_j G(x)e_j = g(x)+\delta_j G_j$$

其中，$G_j$ 为 Hessian 阵的第 $j$ 列，$e_j$ 是第 $j$ 个坐标方向上的单位向量。因此$G_j$ 的前向差分近似为：

$$\tilde{G}_j = \frac{g(x+\delta_j e_j)}{\delta_j} \qquad \text{（式 2.14）}$$

上式对于二次函数是精确的。

重复使用（式 2.14），可以构造出 Hessian 阵$G$ 的近似$\tilde{G}$。一般说来，由于误差，$\tilde{G}$ 不是对称的，为了较好地做 Hessian 阵的近似，特取

$$\hat{G}=\frac{1}{2}\left(\tilde{G}+\tilde{G}^T\right)$$

用$\hat{G}_k$ 来代替原牛顿法的$G_k$，这就是离散牛顿法。如果$\hat{G}_k$ 非正定，则和牛顿法一样，采用拟牛顿法来解决。

#### 2.3.1.5 信赖域法

前面讨论了无约束最优化方法，它们的基本原理是利用二次模型产生下降方向，然后由线性搜索在该方向上选择一个可接受的补偿值，从而得到一个新的迭代点。尽管这种算法已在实践中被证明是有效的，但他们有一个缺点，就是这种二次模型有时并不充分近似于原来的目标函数，因而在基于该二次模型所确定的搜索方向上，常常无法找到一个有满意下降值的迭代点。而信赖域法的基本步骤是首先指定一个最大步长，然后用约束的二次模型来确定惟一的方向与大小。这个最大步长提供了一个使该二次模型成为$f(x)$可信赖域近似模型的区域，信赖域法因此得名。许多研究者认为，这种方法将提供非常有效、并具有十分精致的整体收敛性质的最优化算法

假设在第 $k$ 次迭代开始时已有点$x^{(k)}$与最大步长$\Delta_k$，以及$\nabla^2 f\left(x^{(k)}\right)$的某种近似$B_k$，$B_k$ 为一个对称正定矩阵，于是

$$q(s)=\frac{1}{2}s^T B_k s+\nabla f\left(x^{(k)}\right)^T s+f\left(x^{(k)}\right)$$

就是 $f\left(x^{(k)}+s\right)$ 的某种二次近似。考虑 $q(s)$ 在 $\|s\|\leqslant\Delta_k$ 有极小点 $s^{(k)}$（这里 $\|\bullet\|$ 为某种向量模），也就是说，$s^{(k)}$ 是有约束的、以 $q(s)$ 为目标函数的极小化问题

$$\min_{\|s\|\leqslant\Delta_k} q(s)$$

的最优解。区域 $\|s\|\leqslant\Delta_k$ 称为信赖域，令

$$x^{(k+1)}=x^{(k)}+s^{(k)}$$

于是

$$\Delta f=f\left(x^{(k)}\right)-f\left(x^{(k+1)}\right)$$

就是目标函数 $f(x)$ 的实际下降数值，而

$$\Delta q=f\left(x^{(k)}\right)-q\left(s^{(k)}\right)=q(0)-q\left(s^{(k)}\right)(>0)$$

则是 $f(x)$ 的预测下降数值。根据这两者的比值，可以修正 $\Delta_k$ 得到 $\Delta_{k+1}$。具体地说，如果

$$\Delta f>\frac{3}{4}\Delta q$$

说明在信赖域内，$q(s)$ 相当好地近似于 $f\left(x^{(k)}+s\right)$，因此建议扩大信赖区域，而取

$$\Delta_{k+1}=2\Delta_k$$

如果

$$\Delta f<\frac{1}{10}\Delta q$$

成立，相反建议缩小信赖区域，而取

$$\Delta_{k+1}=\frac{1}{2}\Delta_k$$

如果

$$\frac{1}{10}\Delta q\leqslant\Delta f\leqslant\frac{3}{4}\Delta q$$

则取

$$\Delta_{k+1}=\Delta_k$$

从而可以开始下一步迭代，这就是信赖域法，具体步骤为：

（1）选取初始点 $\boldsymbol{x}^{(1)}$ 和初始矩阵 $\boldsymbol{B}_1$，取定 $\Delta_1$， $k=1$；

（2）求解 $\min\limits_{\|s\|\le\Delta_k} q(\boldsymbol{s})$，得解 $\boldsymbol{s}^{(k)}$；

（3）取 $\boldsymbol{x}^{(k+1)}=\boldsymbol{x}^{(k)}+\boldsymbol{s}^{(k)}$，检查停机准则；

（4）计算

$$\Delta f=f\left(\boldsymbol{x}^{(k)}\right)-f\left(\boldsymbol{x}^{(k+1)}\right),\quad \Delta q=f\left(\boldsymbol{x}^{(k)}\right)-q\left(\boldsymbol{s}^{(k)}\right)$$

而取

$$\Delta_{k+1}=\begin{cases}\dfrac{1}{2}\Delta_k, & \Delta f<0.1\Delta q\\ \Delta_k, & 0.1\Delta q\le\Delta f\le 0.75\Delta q\\ 2\Delta_k, & \Delta f>0.75\Delta q\end{cases}$$

（5）$\boldsymbol{B}_{k+1}$ 代替 $\boldsymbol{B}_k$；

（6）$k=k+1$，转（2）。

## 2.3.2　优化工具箱函数的选用

在 MATLAB 6.5 优化工具箱中，用于求解无约束非线性规划的函数有：fminsearch 和 fminunc，用法介绍如下。

### 2.3.2.1　fminsearch 函数

**【语法】**

```
x = fminsearch (fun, x0)
x = fminsearch (fun, x0, options)
x = fminsearch (fun, x0, options, P1, P2, ...)

[ x, fval ] = fminsearch (...)
[ x, fval, exitflag ] = fminsearch (...)
[ x, fval, exitflag, output ] = fminsearch (...)
```

**【说明】**

fun：是目标函数；

options：设置优化选项参数；

fval：返回目标函数在最优解 $x$ 点的函数值；
exitflag：返回算法的终止标志；
output：返回优化算法信息的一个数据结构。

2.3.2.2 fminunc 函数

**【语法】**

```
x = fminunc (fun, x0)
x = fminunc (fun, x0, options)
x = fminunc (fun, x0, options, P1, P2, ...)

[ x, fval ] = fminunc (...)
[ x, fval, exitflag ] = fminunc (...)
[ x, fval, exitflag, output ] = fminunc (...)
[ x, fval, exitflag, output, grad ] = fminunc (...)
[ x, fval, exitflag, output, grad, hessian ] = fminunc (...)
```

**【说明】**

fun：是目标函数；
options：设置优化选项参数；
fval：返回目标函数在最优解 $x$ 点的函数值；
exitflag：返回算法的终止标志；
output：返回优化算法信息的一个数据结构；
grad：返回目标函数在最优解 $x$ 点的梯度；
hessian：返回目标函数在最优解 $x$ 点的 Hessian 矩阵值。

## 2.3.3 工程应用举例

### 2.3.3.1 fminsearch 函数

**【工程应用背景】**

已知函数 $f(\boldsymbol{x})$，求 $\min\limits_{x} f(x)$。其中 $\boldsymbol{x}=[x_1, x_2, x_3, \ldots, x_n]^T$ 为向量，$f$ 返回一标量值。

**【实例分析】**

求函数 $f(x)=\sin(x)+3$ 取最小值时的 $x$ 值。
例程 2-2 是求解的 MATLAB 代码。

**例程 2-2**

```
%首先编写 f(x)的.m 文件
function f = myfun(x)
 f=sin(x)+3;
%然后调用函数 fminsearch
x0=2   %起始点
```

```
[x , fval]=fminsearch(@myfun, x0);
```

【结果输出】

```
x =
        4.7124
fval=
        2.0000
```

### 2.3.3.2 fminunc 函数

【工程应用背景】

已知函数$f(x)$，求$\min\limits_{x} f(x)$，其中$x = [x_1, x_2, x_3, \cdots x_n]^T$为向量，$f$返回一个标量值。

【实例分析】

求函数$f(x)=3x_1^2+2x_1x_2+x_2^2$的最小值。

例程 2-3 是求解的 MATLAB 代码。

例程 2-3

```
%首先编写 f(x)的.m 文件
function f = myfun (x)
 f = 3*x(1)^2+2*x(1)*x(2)+x(2)^2;
%然后调用函数 fminunc
%起始点
x0 = [1,1]
[x, fval]=fminunc(@myfun, x0)
```

【结果输出】

```
x =
      1.0e-008 *-0.7914   0.2260
fval=
      1.5722e-016
```




求解无约束条件下多变量函数的最小值，fminsearch 函数和 fminunc 函数意义相同。

## 2.4 约束最优化及其优化工具箱函数选用

绝大部分实际问题都受到某些条件的限制，这些限制条件（约束）常给寻优工作带来很大困难。

### 2.4.1 基本理论

这一部分首先说明解的最优性条件，然后研究几种基本的解法。



2.4.1.1　**最优性条件**

**1．等式约束极小的最优性条件**

对于一般的等式约束问题

$$\begin{aligned}&\min_{x\in R^n} f(\boldsymbol{x})\\ &s.t\quad h_j(\boldsymbol{x})=0,\ j=1,2,\cdots,l\end{aligned} \tag{式 2.15}$$

其中 $f$ 和 $h_j$ 均可微，在最优解 $\boldsymbol{x}^*$ 处，必存在数值 $(u_1^*,u_2^*,\cdots,u_l^*)$，使

$$\nabla f(\boldsymbol{x}^*)=\sum_{j=1}^{l}u_j^*\nabla h_j(\boldsymbol{x}^*)$$

对于问题（式 2.15），若 $f$，$h_j\in C^1$，$j=1,2,\cdots,l$ 且 $\nabla h_1(\boldsymbol{x}^*),\cdots,\nabla h_l(\boldsymbol{x}^*)$ 线性无关，则存在最优解的必要条件为：存在相应的拉格朗日乘子 $\boldsymbol{u}^*=(u_1^*,u_2^*,\cdots,u_l^*)^T$，使得

$$\nabla_x L(\boldsymbol{x}^*,\boldsymbol{u}^*)=\nabla f(\boldsymbol{x}^*)-\sum_{j=1}^{l}u_j^*\nabla h_j(\boldsymbol{x}^*)=0$$

**2．一般线性规划的最优性条件**

现考虑

$$\begin{aligned}\min\quad & f(x_1,x_2,x_3)\\ s.t\quad & g_1(x_1,x_2,x_3)\geqslant 0\\ & g_2(x_1,x_2,x_3)\geqslant 0\end{aligned} \tag{式2.16}$$

其中 $f,g_1,g_2$ 均可微。设 $\boldsymbol{x}^*=(x_1^*,x_2^*,x_3^*)$ 为（式 2.16）的最优解，且设

$$g_1(x_1^*,x_2^*,x_3^*)=0,\quad g_2(x_1^*,x_2^*,x_3^*)>0$$

由于 $\boldsymbol{x}^*$ 为区域 $\{(x_1,x_2,x_3)|g_1(x_1,x_2,x_3)\geqslant 0\}$ 的内点，所以对于 $\boldsymbol{x}^*$ 而言，约束 $g_2(x_1,x_2,x_3)\geqslant 0$ 实际上并没起作用。于是，$\boldsymbol{x}^*$ 实际上也是问题

$$\begin{aligned}\min\quad & f(x_1,x_2,x_3)\\ s.t\quad & g_1(x_1,x_2,x_3)\geqslant 0\end{aligned} \tag{式 2.17}$$

的最优解，必有 $\lambda_1^*$，使

$$\nabla f\left(\boldsymbol{x}^{*}\right)-\lambda_{1}^{*} \nabla g_{1}\left(\boldsymbol{x}^{*}\right)=0 \qquad （式 2.18）$$

为形式整齐起见，引进 $\lambda_{2}^{*}=0$ 而把（式 2.18）写成

$$\nabla f\left(\boldsymbol{x}^{*}\right)-\lambda_{1}^{*} \nabla g_{1}\left(\boldsymbol{x}^{*}\right)-\lambda_{2}^{*} \nabla g_{2}\left(\boldsymbol{x}^{*}\right)=0 \qquad （式 2.19）$$

而把 $g_{1}\left(x_{1}^{*}, x_{2}^{*}, x_{3}^{*}\right)=0$ 和 $\lambda_{2}^{*}=0$ 统一写成

$$\lambda_{i}^{*} g_{i}\left(x_{1}^{*}, x_{2}^{*}, x_{3}^{*}\right)=0,\ \ i=1,2 \qquad （式 2.20）$$

从几何上看，（式 2.18）即上段分析的 $\nabla f\left(\boldsymbol{x}^{*}\right)$ 和 $\nabla g_{1}\left(\boldsymbol{x}^{*}\right)$ 都通过 $\boldsymbol{x}^{*}$ 且应共线。其实，由于 $\boldsymbol{x}^{*}$ 是（式 2.17）的最优解，所以当动点 $\boldsymbol{x}$ 由 $\boldsymbol{x}^{*}$ 出发沿 $g_{1}(\boldsymbol{x})=0$ 上的各方向移动时，目标函数值 $f(\boldsymbol{x})$ 均在增加，不仅如此，而且 $\boldsymbol{x}$ 由 $\boldsymbol{x}^{*}$ 出发往 $g_{1}(\boldsymbol{x}) \geqslant 0$ 的内部移动时，$f(\boldsymbol{x})$ 也应增加。

注意到梯度 $\nabla f\left(\boldsymbol{x}^{*}\right)$ 和 $\nabla g_{1}\left(\boldsymbol{x}^{*}\right)$ 均分别指向函数值 $f(\boldsymbol{x})$ 及 $g_{1}(\boldsymbol{x})$ 增加的方向。因此，$\nabla f\left(\boldsymbol{x}^{*}\right)$ 和 $\nabla g_{1}\left(\boldsymbol{x}^{*}\right)$ 不仅共线，而且应是同方向，也即（式 2.20）中 $\lambda_{1}^{*}$ 应非负而 $\lambda_{2}^{*}=0$，所以可统一写为

$$\lambda_{i}^{*} \geqslant 0,\ \ i=1,2$$

总之，（式 2.16）的最优解 $\boldsymbol{x}^{*}$ 必满足条件（式 2.18）、（式 2.19）、（式 2.20）。

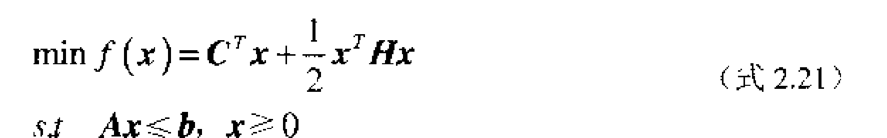
#### 2.4.1.2 二次规划（QP）和序列二次规划（SQP）

由于线性规划和二次规划问题都比较容易求解，所以人们很自然地想到：把要求解的一般非线性约束优化问题线性化，然后用线性规划来逐步求其近似解，这种方法称为线性逼近法或序列线性规划法（SLP）。但是线性逼近法的精度差，收敛速度慢，而近二十年二次规划的解法得到很大发展，有了比较有效的算法，因此近十年来用二次规划的方法即序列二次规划法获得了突出的进展，成为当前世界上最流行的重要的约束优化算法之一，

**1．二次规划（QP 法）**

二次规划当属最简单的非线性规划问题，即约束为线性的而目标函数为二次函数的最优化问题。

考虑二次规划问题



$$\min f(x)=C^{T}x+\frac{1}{2}x^{T}Hx$$
$$s.t \quad Ax\leqslant b,\ x\geqslant 0 \qquad \text{（式 2.21）}$$

其中假定 $m$ 维向量 $b\geqslant 0$，$H$ 为 $n\times n$ 对称半正定矩阵。

我们写出（式 2.21）的拉格朗日函数

$$L(x,u,v)=C^{T}x+\frac{1}{2}x^{T}Hx+u^{T}(Ax-b)-v^{T}x$$

则由最优性条件知，在最优解处，K-T 条件成立

$$\left.\begin{array}{l} Ax+y=b \\ C+Hx+A^{T}u-v=0 \\ x\geqslant 0,u\geqslant 0,v\geqslant 0,y\geqslant 0 \end{array}\right\}$$

$$x^{T}v+u^{T}y=0$$

**2．序列二次规划法（SQP 法）**

考虑一般的约束优化问题

$$\begin{aligned} \min \quad & f(x) \\ s.t \quad & g_i(x)\geqslant 0,\ i=1,2,\cdots,m \\ & h_j(x)=0,\ j=1,2,\cdots,n \end{aligned} \qquad \text{（式 2.22）}$$

SQP 法的基本思想是：把问题（式 2.22）转化为求解一系列的二次规划问题，因此称为序列二次规划法。具体地说，假定在第 $k$ 次迭代开始时已知近似解 $x^{(k)}$ 和近似乘子向量 $\lambda^{(k)}$，根据它们，可以给出第 $k$ 个二次规划子问题 $(P_k)$，求解 $(P_k)$ 即可得到新的近似解 $x^{(k+1)}$，并确定相应的乘子向量 $\lambda^{(k+1)}$，重复上述过程，直到获得问题（式 2.22）所需的近似最优解为止。若令 $d^{(k)}=x^{(k+1)}-x^{(k)}$，则求解问题 $(P_k)$ 来获得 $x^{(k+1)}$ 可转化为求 $d^{(k)}$ 的二次规划子问题 $(P_k)$。

对问题（式 2.22），相应的二次规划问题的约束可以用在点 $x^{(k)}$ 处线性化的方法得到，即应考虑如下的 QP 问题

$$\begin{cases} \min \quad Q(d)=\frac{1}{2}d^TBd+\nabla f\left(x^{(k)}\right)^Td \\ s.t \quad \nabla h_j\left(x^{(k)}\right)^Td+h_j\left(x^{(k)}\right)=0,\ j=1,\cdots,l \\ \nabla g_i\left(x^{(k)}\right)^Td+g_i\left(x^{(k)}\right)\geqslant 0,\ i=1,\cdots,m \end{cases}$$

2.4.1.3　可行方向法

设可行域 $D\in R^n$ 是非空集，$x\in D$，若对于某非零向量 $d\in R^n$，存在使得对任意 $t\in(0,\ \delta)$，均有 $x+td\in D$，则称 $d$ 为从点 $x$ 出发的可行方向。

利用可行方向的概念，典型的策略为：由可行点 $x^{(k)}$ 出发，找一个下降的可行方向 $d^{(k)}$ 作为搜索方向，然后确定沿此方向移动的步长，得到下一迭代点 $x^{(k+1)}$，此类方法称可行方向法。搜索方向的选择不同就形成不同的可行方向法。这里介绍最常用的 Zoutendijk 可行方向法。

考虑

$$\begin{aligned} \min \quad & f(x) \\ s.t \quad & Ax\geqslant b \\ & Gx=c \end{aligned} \qquad \text{（式 2.23）}$$

对于问题（式 2.23），在迭代可行点 $x^{(k)}$ 处，为了得到一个下降可行方向，Zoutendijk 在 1960 年首先提出借助于大家熟悉的线性规划。由于问题（式 2.23）中的约束为线性的，仅目标函数 $f(x)$ 是非线性的，而我们又是在 $x^{(k)}$ 的附近考察 $f(x)$，很自然地想到用 $f(x)$ 在 $x^{(k)}$ 处的一阶泰勒多项式近似代替 $f(x)$。

假设 $x^{(k)}$ 满足 $A_1x^{(k)}>b_1$，$A_2x^{(k)}=b_2$，$Gx^{(k)}=c$，其中 $A=\begin{bmatrix}A_1\\A_2\end{bmatrix}$，$b=\begin{bmatrix}b_1\\b_2\end{bmatrix}$，令 $x=x^{(k)}+d$，$f(x)\approx f\left(x^{(k)}\right)+\nabla f\left(x^{(k)}\right)^Td$。于是得到线性规划



$$
\begin{aligned}
\min \quad & f\left(x^{(k)}\right)+\nabla f\left(x^{(k)}\right)^T d \\
s.t \quad & A_1\left(x^{(k)}+d\right) \geqslant b_1 \\
& A_2\left(x^{(k)}+d\right) \geqslant b_2 \\
& G\left(x^{(k)}+d\right)=c
\end{aligned}
$$

由于 $f\left(x^{(k)}\right)$ 为常数和 $A_1 x^{(k)} > b_1$，所以上述线性规划等价于

$$
\begin{aligned}
\min \quad & \nabla f\left(x^{(k)}\right)^T d \\
s.t \quad & A_2 d \geqslant 0 \\
& Gd = 0
\end{aligned}
\qquad \text{（式 2.24）}
$$

综上所述，Zoutendijk 可行方向法的具体步骤为：

（1）任选可行点 $x^{(0)}$ 做初始点，令 $k=0$；

（2）根据 $x^{(k)}$ 的有效约束把 $A$ 分解为 $A_1$ 和 $A_2$，相应地把 $b$ 分解为 $b_1$ 和 $b_2$，使 $A_1 x^{(k)} > b_1$，$A_2 x^{(k)} = b_2$：

（3）求解线性规划：

$$
\begin{aligned}
\min \quad & \nabla f\left(x^{(k)}\right)^T d \\
s.t \quad & A_2 d \geqslant 0 \\
& Gd = 0 \\
& -1 \leqslant d_i \leqslant 1, \ i=1,\cdots,n
\end{aligned}
$$

设其最优解 $d^*$，限制 $-1 \leqslant d_i \leqslant 1$ 既不妨碍我们获得可行下降方向的目的，又防止线性规划无最优解情况的发生，它类似于信赖域的作用。

（4）若 $\left|\nabla f\left(x^{(k)}\right)^T d^*\right| < \varepsilon$，则 $x^{(k)}$ 即为所求的最优点，停机，否则转（5）；

（5）若 $A_1 d^* \geqslant 0$，则线性搜索 $\min f\left(x^{(k)}+td^*\right)$ 得 $x^{(k+1)}=x^{(k)}+t_k d^*$，令 $k=k+1$ 转（2），否则转（6）；

（6）设 $u = A_1 x^{(k)} - b_1$，$v = A_2 d^*$，计算

$$\bar{t}=\min\left\{-\frac{u_i}{v_i}\middle| v_i<0\right\}$$

由 $\min\limits_{0\le t\le \bar{t}} f\left(\boldsymbol{x}^{(k)}+t\boldsymbol{d}^*\right)$ 确定 $\boldsymbol{x}^{(k+1)}=\boldsymbol{x}^{(k)}+t_k\boldsymbol{d}^*$，令 $k=k+1$，转（2）。

2.4.1.4　惩罚函数法

考虑一般约束最优化问题，本方法的基本策略就是根据约束的特点构造某种“惩罚”项，然后把它加到目标函数中去，使得约束问题的求解转化为一系列无约束问题的求解。这种惩罚策略对于无约束问题求解过程中企图违反约束的那些迭代点，以很大的目标数值迫使这一系列无约束问题的极小点或者不断地向可行域靠近，或者一直保持在可行域内移动，直到收敛于原约束问题的极小点。

一般惩罚函数法有三种方法：第一种是外部惩罚函数法（外点法），它对违反约束的点在目标函数中加入相应的惩罚，而对可行点不予惩罚。此法的迭代点在可行域外部移动；第二种是内部函数惩罚法（内点法），它企图从内部穿越可行域边界的点在目标函数中加入相应的“障碍”，距离边界越近，障碍越大，在边界上给以无穷大的障碍，从而保证迭代一直在可行域内部移动；第三种是乘子法，即在拉格朗日函数内加入相应的惩罚。

1. 外点法

对于等式约束问题，定义辅助函数

$$F_1(\boldsymbol{x},s)=f(\boldsymbol{x})+s\sum_{j=1}^{l}h_j^2(\boldsymbol{x})\qquad\text{（式 2.25）}$$

参数 $\sigma$ 是很大的正数。当我们求解无约束问题

$$\min_{x\in R^n}F_1(\boldsymbol{x},\boldsymbol{s})\qquad\text{（式 2.26）}$$

得到的最优解 $\boldsymbol{x}^*$，满足 $h_j\left(\boldsymbol{x}^*\right)=0,\ j=1,\cdots,l$ 时，显然 $\boldsymbol{x}^*$ 即为原约束问题（式 2.24）的最优解。而在求解问题的过程中，若 $\boldsymbol{x}^{(k)}$ 不满足 $h_j\left(\boldsymbol{x}^{(k)}\right)=0$，则（式 2.25）中的第二项将是很大的正数，限制它成为极小点，以迫使无约束问题（式 2.26）的最优解满足（或接近满足）$h_j\left(\boldsymbol{x}^*\right)=0,\ j=1,\cdots,l$。由此可见，求解问题（式 2.26）能够得到问题（式 2.24）的近似解。

对于不等式约束问题

$$\begin{aligned}&\min\quad f(\boldsymbol{x})\\&s.t\quad\ g_i(\boldsymbol{x})\geqslant 0,\ i=1,2,\cdots,m\end{aligned}$$

可以用同样的思想构造辅助函数，如定义

$$F_2(x,s)=f(x)+s\sum_{j=1}^{l}\left[\max\left\{0,-g_i(x)\right\}\right]^2$$

这样可以把问题转化为无约束问题

$$\min F_2(x,\sigma)$$

对于一般的情形，可定义辅助函数

$$F(x,\sigma)=f(x)+\sigma P(x)$$

其中 $P(x)$ 可取如下形式

$$P(x)=\sum_{i=1}^{m}\left[\max\left\{0,-g_i(x)\right\}\right]^{\alpha}+\sum_{j=1}^{l}\left|h_j(x)\right|^{\beta}$$

其中，$\alpha$，$\beta\geqslant 1$ 均为常数。

这样把约束问题转化为无约束问题

$$\min F(x,\sigma)\overset{def}{=\!=}f(x)+\sigma P(x)$$

2. 内点法

内点法总是从可行域的内点出发，并保证在可行域内部进行搜索。因此这种方法适用于下列只有不等式约束的问题：

$$\begin{aligned}&\min f(x)\\&s.t\quad g_i(x)\geqslant 0,\ \ i=1,2,\cdots,m\end{aligned}\qquad\text{（式 2.27）}$$

现将可行域记为

$$D=\left\{x\,\middle|\,g_i(x)\geqslant 0,\ \ i=1,\cdots,m\right\}$$

保持迭代点含于 $D$ 内部的方法是定义障碍函数

$$G(x,r)=f(x)+rB(x)$$

其中 $B(x)$ 是连续函数，当点 $x$ 趋向边界时，$G(x,r)\to+\infty$；否则，由于 $r$ 很小，则 $G(x,r)\approx f(x)$。因此，我们可以通过求解问题

$$\begin{aligned}&\min G(x,r)\\&s.t\quad x\in \operatorname{int}D\end{aligned}$$

其中符号 $\text{int}\,D$ 表示 $D$ 的内部，得到问题（式 2.27）的近似解。

内点法的具体计算步骤为：

（1）给定初始点 $\boldsymbol{x}^{(0)} \in \text{int}\,D$，允许误差 $\varepsilon > 0$，初始参数 $r_1 > 0$，缩小系统 $\beta \in (0,1)$，$k = 1$；

（2）以 $\boldsymbol{x}^{(k-1)}$ 为初始点，求解问题

$$\begin{aligned} &\min f(\boldsymbol{x}) + r_k B(\boldsymbol{x}) \\ &s.t \quad \boldsymbol{x} \in \text{int}\,D \end{aligned}$$

设求得极小点 $\boldsymbol{x}^{(k)}$；

（3）若 $r_k B\left(\boldsymbol{x}^{(k)}\right) < \in$，则停，得近似解 $\boldsymbol{x}^{(k)}$．否则令 $r_{k+1} = \beta r_k$，$k = k+1$，回（2）。

## 2.4.2 优化工具箱函数的选用

在 MATLAB 6.5 优化工具箱中，用于求解约束最优化问题的函数有：fminbnd、fmincon、fsemcnf、quadprog、fminmax，用法介绍如下。

### 2.4.2.1 fminbnd 函数

【语法】

```
x = fminbnd (fun, x1, x2)
x = fminbnd (fun, x1, x2,options)
x = fminbnd (fun, x1, x2,options, P1, P2, ...)

[x, fval] = fminbnd (...)
[x, fval, exitflag] = fminbnd (...)
[x, fval, exitflag, output] = fminbnd (...)
```

【说明】

fun：是目标函数；

x1, x2：设置优化变量给定区间的上下界；

options：设置优化选项参数；

fval：返回目标函数在最优解 $x$ 点的函数值；

exitflag：返回算法的终止标志；

output：是一个返回优化算法信息的结构。



2.4.2.2　fmincon 函数

【语法】

x = **fmincon** (*fun, x0, A, b*)
x = **fmincon** ( fun, x0, A, b, Aeq, beq)
x = **fmincon** ( fun, x0, A, b, Aeq, beq, lb, ub)
x = **fmincon** ( fun, x0, A, b, Aeq, beq, lb, ub, nonlcon)
x = **fmincon** ( fun, x0, A, b, Aeq, beq, lb, ub, nonlcon, options)
x = **fmincon** ( fun, x0, A, b, Aeq, beq, lb, ub, nonlcon, options, P1, P2, …)

[x, fval] = **fmincon** (…)
[x, fval, existflag] = **fmincon** (…)
[x, fval, existflag, output] = **fmincon** (…)
[x, fval, existflag, output, lambda] = **fmincon** (…)
[x, fval, existflag, output, lambda, grad] = **fmincon** (…)
[x, fval, existflag, output, lambda, grad, hessian] = **fmincon** (…)

【说明】

fun：是目标函数；
options：设置优化选项参数；
fval：返回目标函数在最优解 $x$ 点的函数值；
exitflag：返回算法的终止标志；
output：返回优化算法信息的一个数据结构；
grad：返回目标函数在最优解 $x$ 点的梯度；
hessian：返回目标函数在最优解 $x$ 点的 Hessian 矩阵值。

2.4.2.3　fseminf 函数

【语法】

x = **fseminf** (fun, x0, ntheta, seminfcon)
x = **fseminf** (fun, x0, ntheta, seminfcon, A, b)
x = **fseminf** (fun, x0, ntheta, seminfcon, A, b, Aeq, beq)
x = **fseminf** (fun, x0, ntheta, seminfcon, A, b, Aeq, beq, lb, ub)
x = **fseminf** (fun, x0, ntheta, seminfcon, A, b, Aeq, beq, lb, ub, options)
x = **fseminf** (fun, x0, ntheta, seminfcon, A, b, Aeq, beq, lb, ub, options, P1, P2,…)

[ x, fval ] = **fseminf** (…)
[ x, fval, exitflag ] = **fseminf** (…)
[ x, fval, exitflag , output] = **fseminf** (…)
[ x, fval, exitflag , output, lambda] = **fseminf** (…)

【说明】

fun：是目标函数；
seminfcon：计算非线性约束、线性约束和半无穷约束的函数，它是一个.m 文件或者.mex 文件的文件名；
options：设置优化选项参数；

fval：返回目标函数在最优解 $x$ 点的函数值；
exitflag：返回算法的终止标志；
output：返回优化算法信息的一个数据结构。

#### 2.4.2.4 quadprog 函数

**【语法】**

```
x = quadprog (H, f, A, b)
x = quadprog (H, f, A, b, Aeq, beq)
x = quadprog (H, f, A, b, Aeq, beq, lb, ub)
x = quadprog (H, f, A, b, Aeq, beq, lb, ub, x0)
x = quadprog (H, f, A, b, Aeq, beq, lb, ub, x0, options)
x = quadprog (H, f, A, b, Aeq, beq, lb, ub, x0, options, P1, P2, ...)

[ x, fval ] = quadprog (...)
[ x, fval, exitflag ] = quadprog (...)
[ x, fval, exitflag, output ] = quadprog (...)
[ x, fval, exitflag, output, lambda ] = quadprog (...)
```

**【说明】**

f：设置目标函数中 $x$ 的系数矩阵；
H：设置目标函数中的二次系数矩阵；
options：设置优化选项参数；
fval：返回目标函数在最优解 $x$ 点的函数值；
exitflag：返回算法的终止标志；
output：返回优化算法信息的一个数据结构。

#### 2.4.2.5 fminmax 函数

**【语法】**

```
x = fminimax (fun, x0 )
x = fminimax (fun, x0, A, b )
x = fminimax (fun, x0, A, b, Aeq, beq )
x = fminimax (fun, x0, A, b, Aeq, beq, lb, ub )
x = fminimax (fun, x0, A, b, Aeq, beq, lb, ub, nonlcon)
x = fminimax (fun, x0, A, b, Aeq, beq, lb, ub, nonlcon, options)
x = fminimax (fun, x0, A, b, Aeq, beq, lb, ub, nonlcon, options, P1, P2, ...)

[ x, fval ] = fminimax (...)
[ x, fval, maxfval] = fminimax (...)
[ x, fval, maxfval, exitflag ] = fminimax (...)
[ x, fval, maxfval, exitflag, output ] = fminimax (...)
[ x, fval, maxfval, exitflag, output, lambda ] = fminimax (...)
```

**【说明】**

fun：是目标函数；

options：设置优化选项参数；
fval：返回目标函数在最优解 $x$ 点的函数值；
exitflag：返回算法的终止标志；
output：返回优化算法信息的一个数据结构；
maxfval：返回在最优解 $x$ 点处的函数族 $\{F_i(x)\}$ 中的最大函数值；
grad：返回目标函数在最优解 $x$ 点的梯度；
hessian：返回目标函数在最优解 $x$ 点的 Hessian 矩阵值。

### 2.4.3　工程应用举例

这一部分主要介绍求解约束最优化问题的函数：fminbnd、fmincon、fsemcnf、quadprog、fminmax 等，举出一些应用实例。

#### 2.4.3.1　fminbnd 函数

**【工程应用背景】**

已知函数 $f(x)$，求解其在区间 $x1<x<x2$ 内的最小值，即 $\min_x f(x)$。其中 $x1$，$x2$ 均是标量，$f(x)$返回一个标量值。

**【实例分析】**

求函数 $f(x)=(x-3)^2-1$ 在区间（0, 5）内的最小值。

例程 2-4 是求解的 MATLAB 代码。

例程 2-4

```
%首先编写目标函数的.m 文件:
function f=myfun(x)
f = (x-3).^2-1

%调用函数 fminbnd:
[x,fval] = fminbnd (@myfun, 0, 5);
```

**【结果输出】**

```
x =
      3.0000
fval=
      -1.0000
```

#### 2.4.3.2　fmincon 函数

**【工程应用背景】**

已知约束条件（式 2.28），求解函数 $f(x)$的极小值，即 $\min_x f(x)$　其中 $\boldsymbol{x}$，$\boldsymbol{b}$，$\boldsymbol{beq}$，

***lb*** 和 ***ub*** 均是向量；***A*** 和 ***Aeq*** 是矩阵；***c***(***x***)和 ***ceq***(***x***)是返回值为向量的函数；***f***(***x***)是一个返回值为标量的函数；而且 ***c***(***x***)，***ceq***(***x***)和 ***f***(***x***)可以是非线性函数。

$$\begin{cases} c(x) < 0 \\ ceq(x) = 0 \\ A * x \leqslant b \\ Aeq * x = beq \\ lb \leqslant x \leqslant ub \end{cases} \qquad \text{（式 2.28）}$$

**【实例分析】**

计算使函数 $f(x) = -x_1 x_2 x_3$ 取最小值时的 ***x*** 值，x0=[ 10, 10, 10 ]$^T$，约束条件为 0≤$x_1+2x_2+2x_3 \leqslant 27$。

分析：

将约束条件化为两个不等式：

$$-x_1 + 2 * x_2 + 2 * x_3 \leqslant 0$$
$$x_1 + 2 * x_2 + 2 * x_3 \leqslant 27$$

从而可将它们写成矩阵不等式的形式：

$$A * x \leqslant b$$

其中 ***A*** = [-1, -2, -2; 1, 2, 2]，***b*** = [0; 27]

例程 2-5 是求解的 MATLAB 代码。

例程 2-5

```
%首先编写目标函数的.m 文件:
function   f = myfun (x)
f = -x(1)*x(2)*x(3);

x0=[10; 10; 10]   %起始点
[x, fval] = fmincon (@myfun, x0, A, b)
```

**【结果输出】**

经过 65 步运算，求得：

```
x =
      24.0000
      12.0000
      12.0000
```

对应的函数值为

```
fval =
      -3.4560e+03
```

2.4.3.3　fseminf 函数

**【工程应用背景】**

已知半无穷约束条件（式 2.29），求 $\min_{x} f(x)$。其中 $\boldsymbol{x}$，$\boldsymbol{b}$，$\boldsymbol{beq}$，$\boldsymbol{lb}$ 和 $\boldsymbol{ub}$ 均是向量；$\boldsymbol{A}$ 和 $\boldsymbol{Aeq}$ 是矩阵；$\boldsymbol{c}(\boldsymbol{x})$，$\boldsymbol{ceq}(\boldsymbol{x})$和 $\boldsymbol{K}_i(\boldsymbol{x},\boldsymbol{\omega}_i)$ 是返回值为向量的函数；$f(\boldsymbol{x})$ 返回一个标量值而且 $\boldsymbol{c}(\boldsymbol{x})$，$\boldsymbol{ceq}(\boldsymbol{x})$ 和 $f(\boldsymbol{x})$ 可以是非线性函数；向量组 $\boldsymbol{K}_i(\boldsymbol{x},\boldsymbol{\omega}_i)\leqslant 0$ 是 $\boldsymbol{x}$ 和附加变量 $\boldsymbol{\omega}_1,\cdots,\boldsymbol{\omega}_n$ 的连续函数，且 $\boldsymbol{\omega}_1,\cdots,\boldsymbol{\omega}_n$ 通常是二维向量。

$$\begin{cases} \boldsymbol{c}(\boldsymbol{x})\leqslant 0 \\ \boldsymbol{ceq}(\boldsymbol{x})=0 \\ \boldsymbol{A}*\boldsymbol{x}\leqslant \boldsymbol{B} \\ \boldsymbol{Aeq}*\boldsymbol{x}=\boldsymbol{beq} \\ \boldsymbol{lb}\leqslant \boldsymbol{x}\leqslant \boldsymbol{ub} \\ \boldsymbol{K}_1(\boldsymbol{x},\boldsymbol{\omega}_1)\leqslant 0 \\ \boldsymbol{K}_2(\boldsymbol{x},\boldsymbol{\omega}_2)\leqslant 0 \\ \quad\vdots \\ \boldsymbol{K}_n(\boldsymbol{x},\boldsymbol{\omega}_n)\leqslant 0 \end{cases} \qquad \text{（式 2.29）}$$

**【实例分析】**

求函数 $f(x)=(x_1-0.5)^2+(x_2-0.5)^2+(x_3-0.5)^2$ 取最小值时的 $\boldsymbol{x}$ 值，约束为：

$$\boldsymbol{K}_1(\boldsymbol{x},\boldsymbol{w}_1)=\sin(w_1x_1)\cos(w_1x_2)-\frac{1}{1000}(w_1-50)^2-\sin(w_1x_3)-x_3\leqslant 1$$

$$\boldsymbol{K}_2(\boldsymbol{x},\boldsymbol{w}_2)=\sin(w_2x_2)\cos(w_2x_1)-\frac{1}{1000}(w_2-50)^2-\sin(w_2x_3)-x_3\leqslant 1$$

$$1\leqslant \boldsymbol{w}_1\leqslant 100, \qquad 1\leqslant \boldsymbol{w}_2\leqslant 100$$

分析：

为了将半无穷约束条件化为 $\boldsymbol{K}_i(\boldsymbol{x},\boldsymbol{\omega}_i)\leqslant 0$ 形式，需将给出的约束条件化为

$$\boldsymbol{K}_i(\boldsymbol{x},\boldsymbol{\omega}_i)-1\leqslant 0$$

例程 2-6 是求解的 MATLAB 代码。

**例程 2-6**

```
%首先编写目标函数的.m 文件
```

```
function   f = myfun (x, s)
f = sum( (x - 0.5).^2 );

%然后编写描述非线性等式和不等式约束，以及半无穷约束的函数
function [c, ceq, K1, K2, s] = mycon (X, s)
 if isnan (s(1,1))
    s = (0.2, 0 ;0.2, 0); % 初始化取样步长
 end

 %取样值
 w1 = 1 : s(1, 1) : 100;
 w2 = 1 : s(2, 1) : 100;
 %半无穷约束
 K1=sin(w1*X(1)).*cos(w1*X(2))-1/1000*(w1-50).^2-sin(w1*X(3))-X(3)-1;
 K2=sin(w2*X(2)).*cos(w2*X(1))-1/1000*(w2-50).^2-sin(w2*X(3))-X(3)-1;
 c = [ ]; ceq = [ ];
 %画出半无穷约束的图形
 plot(w1, K1, '-', w2, K2, '+');
 title('semi-infinite constraint');

 %调用函数 fseminf
 x0 = [0.5; 0.2; 0.3]; %初值
 [x, fval] = fseminf (@myfun, x0. 2, @mycon);
```

**【结果输出】**

经过 8 次迭代后，得出：

```
x =
      0.6673
      0.3013
      0.4023
fval =
      0.0770
```

#### 2.4.3.4 quadprog 函数

**【工程应用背景】**

已知条件（式 2.30），求 $\min\limits_{x}\frac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x}+\boldsymbol{f}^T\boldsymbol{x}$。其中 $\boldsymbol{f}$、$\boldsymbol{x}$、$\boldsymbol{b}$、$\boldsymbol{beq}$、$\boldsymbol{lb}$ 和 $\boldsymbol{ub}$ 均是向量；$\boldsymbol{H}$，$\boldsymbol{A}$ 和 $\boldsymbol{Aeq}$ 是矩阵。

$$\begin{cases} \boldsymbol{A}*\boldsymbol{x}\leqslant \boldsymbol{b} \\ \boldsymbol{Aeq}*\boldsymbol{x}=\boldsymbol{beq} \\ \boldsymbol{lb}\leqslant \boldsymbol{x}\leqslant \boldsymbol{ub} \end{cases} \quad \text{（式 2.30）}$$

**【实例分析】**

求使函数 $f(x)=\frac{1}{2}x_1^2+x_2^2-x_1x_2-2x_1-6x_2$ 取最小值时的 $x$ 值，且满足约束条件：

$$\begin{cases} x_1+x_2 \leqslant 2 \\ -x_1+2x_2 \leqslant 2 \\ 2x_1+x_2 \leqslant 3 \\ x_1 \geqslant 0, x_2 \geqslant 0 \end{cases}$$

分析：

首先函数 $f(x)$ 可以写成 $f(x)=\frac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x}+\boldsymbol{f}^T\boldsymbol{x}$ 形式，其中

$$\boldsymbol{H}=\begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix}, \qquad \boldsymbol{f}=\begin{bmatrix} -2 \\ -6 \end{bmatrix}, \qquad \boldsymbol{x}=\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

例程 2-7 是求解的 MATLAB 代码。

例程 2-7

```
H = [1, -1; -1, 2];
f = [-2; -6];
A = [1, 1; -1, 2; 2, 1];
b = [2; 2; 3];
lb = zeros(2, 1);
[x, fval, exitflag, output, lambda] = quadprog (H, f, A, b, [], [], lb);
```

**【结果输出】**

```
x =
        0.6667
        1.3333
fval =
        -8.2222
exitflag =
        1
output =
        iterations: 3
        algorithm: 'medium-scale: active-set'
        firstorderopt: []
        cgiterations: []
lambda =
        lower: [2x1 double]
        upper: [2x1 double]
        eqlin: [0x1 double]
        ineqlin: [3x1 double]
```

2.4.3.5 fminmax 函数

【工程应用背景】

已知约束条件（式 2.31），求解 $\min\limits_{x}\max\limits_{\{F_i\}} F_i(x)$，其中 $x$，$b$，$beq$，$lb$ 和 $ub$ 均是向量；$A$ 和 $Aeq$ 是矩阵；$c(x)$, $ceq(x)$和 $F(x)$是返回值为向量的函数；而且 $c(x)$, $ceq(x)$ 和 $F(x)$ 可以是非线性函数。

$$\begin{cases} c(x)\leqslant 0 \\ ceq(x)=0 \\ A*x\leqslant b \\ Aeq*x=beq \\ lb\leqslant x\leqslant ub \end{cases} \tag{式 2.31}$$

【实例分析】

求函数族$\{f_1(x), f_2(x), f_3(x), f_4(x), f_5(5)\}$取极大极小值时的 $x$ 值。其中：

$$\begin{cases} f_1(x)=2x_1^{\ 2}+x_2^{\ 2}-48x_1-40x_2+304 \\ f_2(x)=-x_1^{\ 2}-3x_2^{\ 2} \\ f_3(x)=x_1+3x_2-18 \\ f_4(x)=-x_1-x_2 \\ f_5(x)=x_1+x_2-8 \end{cases}$$

例程 2-8 是求解的 MATLAB 代码。

例程 2-8

```
%首先编写函数族 {f_i(x)} 的.m 文件
function   f = myfun (x)
f(1)=2*x(1)^2+x(2)^2-48*x(1)-40*x(2)+304;
f(2)=-x(1)^2-3*x(2)^2;
f(3)=x(1)+3*x(2)-18;
f(4)=-x(1)-x(2);
f(5)=x(1)+x(2)-8;

%然后调用函数 fminimax
x0=[0.1, 0.1]; %起始点
[ x, fval ]=fminimax(@myfun, x0);
```

【结果输出】

经过 7 迭代运算，求得：

x =

```
        4.0000
        4.0000
对应的函数值为:
  fval =
        0.0000   -64.0000   -2.0000   -8.0000   -0.0000
```

# 2.5 多目标规划及其优化工具箱函数选用

早在 1772 年，Franklin 就提出了多目标问题矛盾如何协调的问题；1838 年，Cournot 从经济学角度提出了多目标问题的模型；1896 年，Pareto 首次从数学角度提出了多目标最优决策问题；1944 年，von Neumann 从对策论角度奠定了经济行为理论的基础；1951 年，Koopmans 在生产和分配问题中提出有效向量的概念，与此同时，Kuhn 等人给出了向量极值问题有效解的必要条件；1953 年，Arrow 等人又提出了有效点的概念。至此，多目标规划逐渐受到人们的关注，从 20 世纪 50 年代末到 20 世纪 60 年代末，Charnes、Karlin、Zadeh、Klinger、Polak、Keeney 和 Geoffrion 等人先后做出了较有影响的工作，自 1972 年 Yu 提出了支配结构等重要概念后，多目标规划理论的研究更为引人注目，越来越多的人致力于把多目标决策作为工具去解决经济、管理、工程、军事和社会等领域中出现的复杂问题。而且多目标规划理论中的不少问题不仅刺激了运筹学中其他分支的研究，也为这些分支的理论研究提供了新的思想、新的方法和广泛的应用前景。可以毫不夸张地说，多目标决策已成为运筹学的重要分支之一。

## 2.5.1 基本理论

多目标规划的一般形式为：

$$(vp)\begin{cases}\min\left(f_1(\boldsymbol{x}),f_2(\boldsymbol{x}),\cdots,f_p(\boldsymbol{x})\right)^T, & p>1,\ \boldsymbol{x}\in \boldsymbol{E}^p\\ s.t\quad g_i(\boldsymbol{x})\geq 0, & i=1,2,\cdots,m\\ \quad\quad h_i(\boldsymbol{x})=0, & i=1,2,\cdots,l\end{cases} \tag{式 2.32}$$

或者记为

$$\min_{x\in D}\boldsymbol{f}(\boldsymbol{x}),\ D=\left\{x\in E^p \middle| \boldsymbol{g}(\boldsymbol{x})\geq 0,\boldsymbol{h}(\boldsymbol{x})=0\right\}$$

其中 $\boldsymbol{f}(\boldsymbol{x})=\left(f_1(\boldsymbol{x}),f_2(\boldsymbol{x}),\cdots,f_p(\boldsymbol{x})\right)^T$，$\boldsymbol{g}(\boldsymbol{x})=\left(g_1(\boldsymbol{x}),\cdots,g_m(\mathbf{x})\right)^T$

$\boldsymbol{h}(\boldsymbol{x})=\left(h_1(\boldsymbol{x}),\cdots,h_l(\boldsymbol{x})\right)^T$。

求解多目标规划的最基本方法为评价函数法。

评价函数法的基本思想是：借助于几何或应用中的直观背景，构造所谓的评价函数，从而将多目标优化问题转化为单目标优化问题。然后利用单目标优化的求解方法求出最优

解，并把这种最优解当做多目标规划的最优解。

下面介绍几种常用的构造评价函数的思路。

2.5.1.1 理想点法

在（$vp$）中，先求解 $p$ 个单目标问题

$$\min f_j(\boldsymbol{x}),\ j=1,\cdots,p$$

设其最优值为 $f_j^*$，我们称 $\boldsymbol{f}^*=\left(f_1^*,\cdots,f_p^*\right)^T$ 为值域中的一个理想点，因为一般很难达到它。这样，就期望在某种度量下，寻求距离 $\boldsymbol{f}^*$ 最近的 $\boldsymbol{f}$ 作为近似值。一种最直接的方法是构造评价函数

$$\varphi(z)=\sqrt{\sum_{i=1}^{p}\left(z_i-f_i^*\right)^2}$$

然后极小化 $\varphi\left(\boldsymbol{f}(\boldsymbol{x})\right)$，即求解

$$\min_{x\in D}\varphi\left(\boldsymbol{f}(\boldsymbol{x})\right)=\sqrt{\sum_{i=1}^{p}\left(f_i(\boldsymbol{x})-f_i^*\right)^2}$$

并将它的最优解 $\boldsymbol{x}^*$ 作为（$vp$）在这种意义下的最优解。

2.5.1.2 线性加权和法

在具有多个指标的问题中，人们总希望对那些相对重要的指标给予较大的权系数。基于这个现实，自然如下所示构造评价函数，令

$$\Lambda^+=\left\{\boldsymbol{\lambda}=\left(\lambda_1,\cdots,\lambda_p\right)^T\middle|\lambda_i\geqslant 0\text{且}\sum_{i=1}^{p}\lambda_i=1\right\}$$

称之为权向量集。令

$$p(z)=\sum_{i=1}^{p}\lambda_i z_i=F,\lambda\in\Lambda^+\text{或}\Lambda^{++}$$

再求解

$$\min_{x\in D}\{f(x)\}=F(x)$$

而将它的最优解 $\boldsymbol{x}^*$ 作为（$vp$）在这种意义下的最优解。



2.5.1.3 极大极小法

在决策时，采取保守策略是稳妥的。即在最坏的情况下，寻求最好的结果。按照这种想法，可以构造下述的评价函数

$$\varphi(z)=\max_{1\le i\le p} z_i$$

然后求解

$$\min_{x\in D}\varphi(f(x))=\min_{x\in D}\max_{1\le i\le p} f_i(x)$$

并将它的最优解 $x^*$ 作为（$vp$）在这种意义下的最优解。

## 2.5.2 优化工具箱函数的选用

在 MATLAB 6.5 优化工具箱中，用于求解约束最优化问题的函数有：fgoalattain，用法如下。

**【语法】**

```
x = fgoalattain (fun, x0, goal, weight )
x = fgoalattain (fun, x0, goal, weight, A, b)
x = fgoalattain (fun, x0, goal, weight, A, b, Aeq, beq )
x = fgoalattain (fun, x0, goal, weight, A, b, Aeq, beq, lb, ub )
x = fgoalattain (fun, x0, goal, weight, A, b, Aeq, beq, lb, ub, nonlcon )
x = fgoalattain (fun, x0, goal, weight, A, b, Aeq, beq, lb, ub, nonlcon, options )
x = fgoalattain (fun, x0, goal, weight, A, b, Aeq, beq, lb, ub, nonlcon, options, P1, P2, ... )

[x, fval] = fgoalattain (...)
[x, fval, attainfactor] = fgoalattain (...)
[x, fval, attainfactor, existflag] = fgoalattain (...)
[x, fval, attainfactor, existflag, output] = fgoalattain (...)
[x, fval, attainfactor, existflag, output, lambda] = fgoalattain (...)
```

**【说明】**

具体的参数含义如下：

fun：代表要优化的目标函数，它的变量为向量 $\boldsymbol{x}$，返回在 $\boldsymbol{x}$ 点处向量值 $\boldsymbol{F}$。在实际编程中，fun 可以有两种形式：

- 一是 fun 为一个函数句柄，如 $x$ = fgoalattain (@myfun, x0, goal, weight)，其中 myfun 是用户自定义的 MATLAB 函数，即：

```
function     F = myfun (x)
F = ··· %计算函数在 x 点的值。
```

- 二是 fun 为一个内联对象（inline object）。如 $x$ = fgoalattain (inline('sin(x)'), x0, goal, weight) ；

x0：代表优化搜索的起始点；

goal：代表函数 fun 要逼近的目标值，是一个向量，它的维数大小等于目标函数 fun 返回向量 ***F*** 的维数大小。fgoalattain 的优化过程就是使得 ***F*** 逼近 goal；

weight：代表给定的权值向量，用于控制目标逼近过程的步长；

nonlcon：函数由每一个输入变量 $\boldsymbol{x}$ 计算出 $\boldsymbol{c}$ 和 $\boldsymbol{ceq}$，用于估计给定 $\boldsymbol{x}$ 点的非线性不等式 $\boldsymbol{c}(\boldsymbol{x})\leqslant 0$ 和非线性等式 $\boldsymbol{ceq}(\boldsymbol{x})=0$。nonlcon 可以是一个函数句柄，含义同"myfun"。

options：设置优化选项参数，具体意义解释如下。

- Display 设置显示算法返回值的类别，"off"表示不显示结果，"iter"表示每次迭代都显示返回值，"final"表示只显示最后返回值；
- MaxFunEval：设置算法中函数估计的最大数目；
- MaxIter：设置算法迭代的最大次数；
- TolX：设置使算法终止的 $\Delta x$ 值。

fval：返回目标函数在最优解 $\boldsymbol{x}$ 点的函数值；

Exitflag：返回算法的终止标志。exitflag>0 表示算法因求得了最小值而停止；exitflag=0 表示算法因迭代次数越限而停止；exitflag < 0 表示算法因函数值发散而停止；

Output：是一个返回优化算法信息的结构，它的成员 iterations 返回迭代的次数。funcCount 返回算法中函数估计值的数目，algorithm 表示使用的算法。

### 2.5.3 工程应用举例

**【工程应用背景】**

已知：约束条件（式 2.33），求解 $\min\limits_{x,\gamma}\gamma$。其中 ***x***, ***weight***, ***goal***, ***b***, ***beq***, ***lb*** 和 ***ub*** 均是向量；***A*** 和 ***Aeq*** 是矩阵；$\boldsymbol{c}(\boldsymbol{x})$, $\boldsymbol{ceq}(\boldsymbol{x})$和 $\boldsymbol{F}(\boldsymbol{x})$均是返回值为向量的非线性函数。

$$\begin{cases}\boldsymbol{F}(\boldsymbol{x})-\boldsymbol{weight}*\gamma\leqslant \boldsymbol{goal}\\ \boldsymbol{c}(\boldsymbol{x})\leqslant 0\\ \boldsymbol{ceq}(\boldsymbol{x})=0\\ \boldsymbol{A}*\boldsymbol{x}\leqslant \boldsymbol{b}\\ \boldsymbol{Aeq}*\boldsymbol{x}=\boldsymbol{beq}\\ \boldsymbol{lb}\leqslant \boldsymbol{x}\leqslant \boldsymbol{ub}\end{cases} \quad （式 2.33）$$

**【实例分析】**

已知一个线性系统，其状态方程如（式 2.34）所示，外接一个输出反馈控制器 $K$ 构成了一闭环系统，闭环系统的特征值由矩阵 ***A***, ***B***, ***C*** 和 $K$ 决定，即 eig (***A***+***B****$K$****C***)。

$$\boldsymbol{A}=\begin{bmatrix}-0.5 & 0 & 0\\ 0 & -2 & 10\\ 0 & 1 & -2\end{bmatrix},\ \boldsymbol{B}=\begin{bmatrix}1 & 0\\ -2 & 2\\ 0 & 1\end{bmatrix},\ \boldsymbol{C}=\begin{bmatrix}1 & 0 & 0\\ 0 & 0 & 1\end{bmatrix} \quad （式 2.34）$$

分析：



系统要达到的目标为 goal = [-5, -3, -1]，取初始值 $K$ = [-1, -1; -1, -1]；

例程 2-9 是求解的 MATLAB 代码。

例程 2-9

```
%首先编写目标函数的.m 文件
function F = eigfun (K, A, B, C)
F = sort (eig(A+B*K*C))   % 计算目标值

A = [-0.5, 0, 0; 0, -2, 10; 0, 1, -2 ];
B = [ 1, 0; -2, 2; 0, 1 ];
C = [ 1, 0, 0; 0, 0, 1 ];
K0 = [-1, -1,; -1, -1];
goal = [-5, -3, -1];
weight =abs (goal);
lb = -4*ones(size(K0));
ub = 4*ones(size(K0));
options = optimset( 'Display', 'iter');
[K, fval, attainfactor] = fgoalattain(@eigfun, K0, goal, weight,[ ], [ ], [ ], [ ], lb, ub, [ ], options, A, B, C);
```

**【结果输出】**

经过 12 次迭代运算，优化算法停止：

```
K=
        -4.0000    -0.2564
        -4.0000    -4.0000
fval =
         -6.9313
         -4.1588
         -1.4099
attainfactor =
         -0.3863
```

# 2.6　大规模优化问题

当变量 $n$ 相当大时，一般规模的算法会遇到两个困难：一是要求计算时间相当长；另一个就是计算机的存储发生困难，即没有足够的内存来存放计算搜索方向所需要的矩阵。

通常大规模无约束优化问题往往具有特殊的性质，例如问题的海色阵一般是稀疏的。我们把非零元素占整个矩阵的元素数目的比率称为密度。所谓稀疏矩阵就是密度很小的矩阵。实践中，密度是随问题的尺寸的增大而减小的，而非零元素的增加却是线性于 $n$ 的。

在大规模无约束优化问题中，通常预先知道某些变量没有非线性的影响，这就意味着在海色阵中相应元素为零。大规模无约束优化问题的海色阵一般有特殊的结构。所谓的特殊结构是指它的非零元素的分布具有一定的结构形式。根据问题的性质和变量的关系，一般预先知道这种非零元素发生在某些位置上。

### 2.6.1 稀疏的离散牛顿法

当遇到海色阵是稀疏矩阵并且具有一些特殊结构时，离散牛顿法往往是有效的。回忆一下前面叙述的离散牛顿法，我们使用有限差分向量来近似海色阵的各个列。那里需要的梯度赋值在 $n$ 相当大时计算量是很大的。当海色阵是稀疏且对称时，我们可以选择一个专门的有限梯度差分向量，这样使得对 $\boldsymbol{G}$ 的近似只需要很少的梯度赋值。

稀疏问题的离散牛顿法通常由两步组成：第一步是预先处理。即对给定的稀疏矩阵进行行和列的置换，得到一个能减小梯度赋值的典型结构；第二步是用差分矩阵来近似计算稀疏矩阵的近似矩阵 $\hat{\boldsymbol{G}}_k$。要指出的是稀疏海色阵的近似是被用来计算搜索方向的。我们通过一个包含 $\hat{\boldsymbol{G}}_k$ 的稀疏线性系统而获得搜索方向：

$$\hat{\boldsymbol{G}}_k \boldsymbol{p}^k = -\boldsymbol{g}_k$$

### 2.6.2 矩阵 Cholesky 分解和拟牛顿方程求解

拟牛顿法和离散牛顿法都要求求解牛顿方程获得搜索方向 $\boldsymbol{p}^k$：

$$\boldsymbol{B}_k \boldsymbol{p}^k = -\boldsymbol{g}_k$$

可以用矩阵 Cholesky 的分解来解这个牛顿方程。

在求解牛顿方程获得搜索方向 $\boldsymbol{p}^k$ 的过程中，为了保持数值稳定性，我们采用矩阵 Cholesky 的分解方法。如有 $\boldsymbol{B}_k$ 的分解形式：

$$\boldsymbol{B}_k = \boldsymbol{L}_k \boldsymbol{D}_k \boldsymbol{L}_k^T$$

其中 $\boldsymbol{L}_k = \left(l_{ij}^k\right)$ 是对角元素均为 1 的下三角阵，$\boldsymbol{D}_k = \left(d_{ii}^k\right)$ 是对角元素均为正数的对角阵。有：

$$\boldsymbol{L}_k \boldsymbol{D}_k \boldsymbol{L}_k^T \boldsymbol{p}^k = -\boldsymbol{g}_k$$

于是，$\boldsymbol{p}^k$ 可以方便地求得。首先解

$$\boldsymbol{L}_k \cdot \boldsymbol{y}^k = -\boldsymbol{g}_k$$

即

$$y_1^k = -\frac{\partial f\left(x^k\right)}{\partial x_1},$$

和

$$y_i^k = -\frac{\partial f\left(x^k\right)}{\partial x_i} + \sum_{j=1}^{i-1} l_{ij}^k y_j^k, \quad i = 2, \cdots, n$$

再解

$$\boldsymbol{L}_k^T \boldsymbol{p}^k = \boldsymbol{D}_k^{-1} \boldsymbol{y}_k$$

即得

$$p_n^k = y_n^k / d_{nn}^k,$$
$$p_i^k = \frac{y_i^k}{d_{ii}^k} - \sum_{j=i+1}^{n} l_{ji}^k p_j^k, \quad i = n-1, \cdots, 1$$



大规模优化问题优化工具箱函数的选用请参见“第 4 章”。

# 2.7　最小二乘优化及其优化工具箱函数选用

最小二乘优化是一类比较特殊的优化问题，在处理这类问题时，MATLAB 也提供了一些强大的函数支持。

## 2.7.1　基本理论

在无约束最优化问题中，有些重要的特殊情形，比如目标函数由若干个函数的平方和构成。这类函数一般可以写成：

$$F(x) = \sum_{i=1}^{m} f_i^2(x), x \in E^n$$

其中 $x = (x_1, x_2, \ldots, x_n)^{\mathrm{T}}$，一般假设 $m \geqslant n$。我们把极小化这类函数的问题：

$$\min F(x) \overset{\Delta}{=} \sum_{i=1} f_i^2(x)$$

称为最小二乘优化问题。

当 $f_i(x)$ 是 $x$ 的线性函数时，称为线性最小二乘优化问题；当 $f_i(x)$ 是 $x$ 的非线性函

数时，称为非线性最小二乘优化问题。

由于目标函数具有若干个函数平方和这种特殊形式，因此给求解带来方便，除了能够运用一般的求解方法外，还有更简便的解法。MATLAB 6.5 优化工具箱中有专门的求解最小二乘优化问题的函数。

## 2.7.2 优化工具箱函数的选用

在 MATLAB 6.5 优化工具箱中，用于求解最小二乘优化问题的函数有：lsqlin、lsqcurvefit、lsqnonlin、lsqnonneg，用法介绍如下。

### 2.7.2.1 lsqlin 函数

【语法】

x = **lsqlin** (C, d, A, b)
x = **lsqlin** (C, d, A, b, Aeq, beq)
x = **lsqlin** (C, d, A, b, Aeq, beq, lb, ub)
x = **lsqlin** (C, d, A, b, Aeq, beq, lb, ub, x0)
x = **lsqlin** (C, d, A, b, Aeq, beq, lb, ub, x0, options)
x = **lsqlin** (C, d, A, b, Aeq, beq, lb, ub, x0, options, P1, P2, ...)

[x, resnorm ] = **lsqlin**(...)
[x, resnorm, residual] = **lsqlin**(...)
[x, resnorm, residual, exitflag] = **lsqlin**(...)
[x, resnorm, residual, exitflag, output] = **lsqlin**(...)
[x, resnorm, residual, exitflag, output, lambda] = **lsqlin**(...)

【说明】

options：设置优化选项参数；
exitflag：返回算法的终止标志；
output：返回优化算法信息的一个数据结构。

### 2.7.2.2 lsqcurvefit 函数

【语法】

x = **lsqcurvefit** (fun, x0, $x_{data}$, $y_{data}$ )
x = **lsqcurvefit** (fun, x0, $x_{data}$, $y_{data}$, lb, ub)
x = **lsqcurvefit** (fun, x0, $x_{data}$, $y_{data}$, lb, ub, options)
x = **lsqcurvefit** (fun, x0, $x_{data}$, $y_{data}$, lb, ub, options, P1, P2, ...)

[x, resnorm ] = **lsqcurvefit** (...)
[x, resnorm, residual] = **lsqcurvefit** (...)
[x, resnorm, residual, exitflag] = **lsqcurvefit** (...)
[x, resnorm, residual, exitflag, output] = **lsqcurvefit** (...)
[x, resnorm, residual, exitflag, output, lambda] = **lsqcurvefit** (...)
[x, resnorm, residual, exitflag, output, lambda, jacobian] = **lsqcurvefit** (...)



【说明】

fun：是目标函数；

$x_{data}$，$y_{data}$：要拟合的数据；

options：设置优化选项参数；

exitflag：返回算法的终止标志；

output：返回优化算法信息的一个数据结构。

### 2.7.2.3　lsqnonlin 函数

【语法】

```
x = lsqnonlin (fun, x0)
x = lsqnonlin (fun, x0, lb, ub)
x = lsqnonlin (fun, x0, lb, ub, options)
x = lsqnonlin (fun, x0, lb, ub, options, P1, P2, ...)

[x, resnorm ] = lsqnonlin(...)
[x, resnorm, residual] = lsqnonlin(...)
[x, resnorm, residual, exitflag] = lsqnonlin(...)
[x, resnorm, residual, exitflag, output] = lsqnonlin(...)
[x, resnorm, residual, exitflag, output, lambda] = lsqnonlin(...)
[x, resnorm, residual, exitflag, output, lambda, jacobian] = lsqnonlin(...)
```

【说明】

fun：是目标函数；

options：设置优化选项参数；

exitflag：返回算法的终止标志；

output：返回优化算法信息的一个数据结构。

### 2.7.2.4　lsqnonneg 函数

【语法】

```
x = lsqnonneg(C, d)
x = lsqnonneg (C, d, x0)
x = lsqnonneg (C, d, x0, options)

[x, resnorm ] = lsqnonneg (...)
[x, resnorm, residual] = lsqnonneg (...)
[x, resnorm, residual, exitflag] = lsqnonneg (...)
[x, resnorm, residual, exitflag, output] = lsqnonneg (...)
[x, resnorm, residual, exitflag, output, lambda] = lsqnonneg (...)
```

【说明】

options：设置优化选项参数；

exitflag：返回算法的终止标志；

output：返回优化算法信息的一个数据结构。

### 2.7.3 工程应用举例

#### 2.7.3.1 lsqlin 函数

**【工程应用背景】**

在满足约束条件（式 2.35）下，求解 $\min\limits_{x}\frac{1}{2}\|\boldsymbol{Cx}-\boldsymbol{d}\|_2^2$。其中：$\boldsymbol{C}$，$\boldsymbol{Aeq}$，$\boldsymbol{A}$ 均为矩阵；$\boldsymbol{d}$，$\boldsymbol{beq}$，$\boldsymbol{b}$，$\boldsymbol{lb}$，$\boldsymbol{ub}$，$\boldsymbol{x}$ 为向量。

$$\begin{cases}\boldsymbol{A}*\boldsymbol{x}\leqslant\boldsymbol{b}\\ \boldsymbol{Aeq}*\boldsymbol{x}=\boldsymbol{beq}\\ \boldsymbol{lb}\leqslant\boldsymbol{x}\leqslant\boldsymbol{ub}\end{cases} \qquad \text{（式 2.35）}$$

**【实例分析】**

已知 $\boldsymbol{x}$ 的约束条件 $\begin{cases}\boldsymbol{A}*\boldsymbol{x}\leqslant\boldsymbol{b}\\ \boldsymbol{lb}\leqslant\boldsymbol{x}\leqslant\boldsymbol{ub}\end{cases}$，求解 $\min\limits_{x}\frac{1}{2}\|\boldsymbol{C}*\boldsymbol{x}-\boldsymbol{d}\|_2^2$，其中：

$$\boldsymbol{C}=\begin{bmatrix}0.9501 & 0.7620 & 0.6153 & 0.4057\\ 0.2311 & 0.4564 & 0.7919 & 0.9354\\ 0.6068 & 0.0185 & 0.9218 & 0.9169\\ 0.4859 & 0.8214 & 0.7382 & 0.4102\\ 0.8912 & 0.4447 & 0.1762 & 0.8936\end{bmatrix},\quad \boldsymbol{d}=\begin{bmatrix}0.0578\\ 0.3528\\ 0.8131\\ 0.0098\\ 0.1388\end{bmatrix},\quad \boldsymbol{b}=\begin{bmatrix}0.5251\\ 0.2026\\ 0.6721\end{bmatrix}$$

$$\boldsymbol{A}=\begin{bmatrix}0.2027 & 0.2721 & 0.7467 & 0.4659\\ 0.1987 & 0.1988 & 0.4450 & 0.4186\\ 0.6037 & 0.0152 & 0.9318 & 0.8462\end{bmatrix}$$

例程 2-10 是求解的 MATLAB 代码。

例程 2-10

```
A=[0.2027,0.2721,0.7467,0.4659;...
   0.1987,0.1988,0.4450,0.4186;...
   0.6037,0.0152,0.9318,0.8462];
C=[0.9501,0.7620,0.6153,0.4057;...
   0.2311,0.4564,0.7919,0.9354;...
   0.6068,0.0185,0.9218,0.9169;...
   0.4859,0.8214,0.7382,0.4102;...
   0.8912,0.4447,0.1762,0.8936];
d=[0.0578,0.3528,0.8131,0.0098,0.1388]';
b=[0.5251,0.2026,0.6721]';
```



```
lb=-0.1*ones(4,1);
ub=2*ones(4,1);
[x,exitflag,output]=lsqlin(C,d,A,b,[],[],lb,ub)
```

【结果输出】

```
x =
     -0.1000
     -0.1000
      0.2152
      0.3502
exitflag =
      0.1672
output =
      0.0455
      0.0764
     -0.3562
      0.1620
      0.0784
```

2.7.3.2　lsqcurvefit 函数

【工程应用背景】

根据给定输入输出数列 $x_{data}$, $y_{data}$，求出拟合曲线方程 $y=f(x)$，使得

$$\min_x \frac{1}{2}\| F(x, x_{data}) - y_{data} \|_2^2) = \frac{1}{2}\sum_i (F(x, x_{data_i}) - y_{data_i})^2$$

【实例分析】

已知 $x_{data}$ = [3.6, 7.7, 9.3, 4.1, 8.6, 2.8, 1.3, 7.9, 10.0, 5.4 ]，$y_{data}$ = [16.5, 150.6, 263.1, 24.7, 208.5, 9.9, 2.7, 163.9, 325.0, 54.3]，x0=[10, 10, 10]，求解系数 $a$，使得函数 $f(x) = a(1) * x^2 + a(2) * \sin(x) + a(3) * x^3$ 是 $x_{data}$, $y_{data}$ 的最佳拟合曲线。

分析：

由题意，实际上求 $a$，使得满足 $\min_a \frac{1}{2} \| f(a, x_{data}) - y_{data} \|_2^2$

例程 2-11 是求解的 MATLAB 代码。

例程 2-11

```
%首先编写函数 f(x)
function f = myfun (a, x)
f = a(1)*x.^2+a(2)*sin(x)+a(3)*x.^3;

x= [3.6, 7.7, 9.3, 4.1, 8.6, 2.8, 1.3, 7.9, 10.0, 5.4 ];
y= [16.5, 150.6, 263.1, 24.7, 208.5, 9.9, 2.7, 163.9, 325.0, 54.3 ];
a0 = [10, 10, 10];
```

```
[a, residual] = lsqcurvefit(@myfun, a0, x, y);
```

【结果输出】

```
a =
     0.2269   0.3385   0.3021
resnorm =
     6.2950
```

即

$$f(x) = 0.2269 * x^2 + 0.3385 * \sin(x) + 0.3021 * x^3$$

图 2-1（a）表示给定数据（x, y）的分布图，图 2-1（b）画出了拟合后的曲线图。

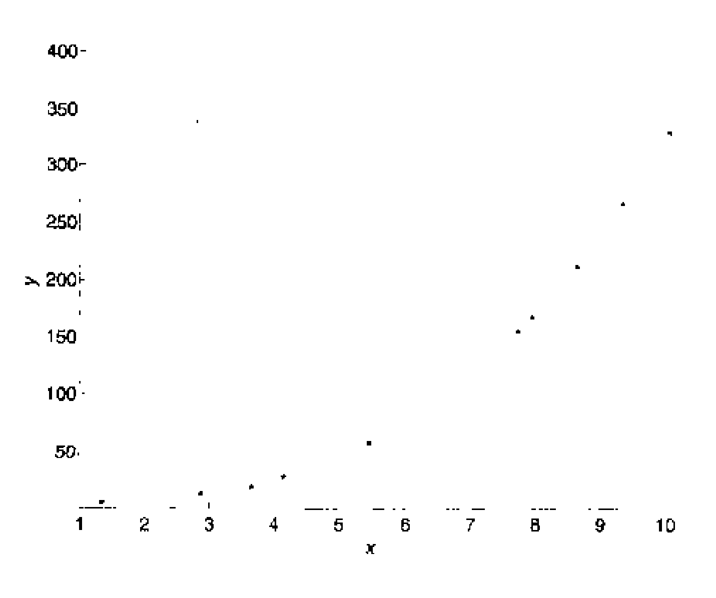



图 2-1（a） 给定数据(x, y)的分布

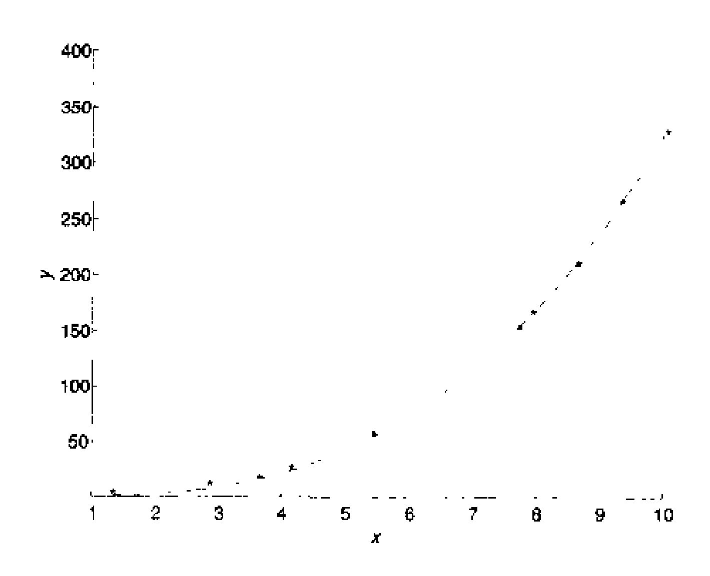



图 2-1（b） 拟合后的曲线

2.7.3.3　lsqnonlin 函数

【工程应用背景】

已知函数矩阵 $F(x)$，$F(x)=\begin{bmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_k(x) \end{bmatrix}$，求 $x$ 使得 $\min\limits_{x}\frac{1}{2}\|F(x)\|_2^2$。

【实例分析】

已知函数向量 $F(x)=\begin{bmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_{10}(x) \end{bmatrix}$，$f_k(x)=2+2k-e^{kx_1}-e^{kx_2}$，求 $x$ 使得 $\min\limits_{x}\frac{1}{2}\|F(x)\|_2^2$。

分析:

由题意可得：$\frac{1}{2}\|F(x)\|_2^2=\sum\limits_{k=1}^{10}f_k(x)^2$

例程 2-12 是求解的 MATLAB 代码。

例程 2-12

```
%首先编写目标函数的.m 文件
function F = myfun (x)
k = 1:10;
F = 2+2*k-exp(k*x(1))-exp(k*x(2));

x0 = [0.3, 0.4];
[x, resnorm] = lsqnonlin (@myfun, x0);
```

【结果输出】

```
x =
        0.2578    0.2578
resnorm =
        124.3622
```

2.7.3.4　lsqnonneg 函数

【工程应用背景】

求解非负的 $x$，使得满足 $\min\limits_{x}\frac{1}{2}\|Cx-d\|_2^2$。

**【实例分析】**

已知：$C=\begin{bmatrix}0.0372 & 0.2869\\ 0.6861 & 0.7071\\ 0.6233 & 0.6245\\ 0.6344 & 0.6170\end{bmatrix}$，$d=\begin{bmatrix}0.8587\\ 0.1781\\ 0.0747\\ 0.8405\end{bmatrix}$，求 $x(x\geqslant 0)$满足$\frac{1}{2}\|Cx-d\|_2^2$最小。

例程 2-13 是求解的 MATLAB 代码。

例程 2-13

```
c=[0.0372,0.6861,0.6233,0.6344;0.2869,0.7071,0.6245,0.6170]';
d = [0.8587, 0.1781, 0.0747, 0.8405]';
[x, resnorm] = lsqnonneg (c, d);
```

**【结果输出】**

```
x =
     0
     0.6929
resnorm =
     0.8315
```

# 2.8　其他函数的工程应用

通过前面几节的介绍，我们已经基本上可以解决比较常见的优化问题了。然而除此之外，MATLAB 还提供了其他一些用于优化的函数，下面我们将对常用的四个函数逐一介绍，并举出例子。

## 2.8.1　方程求解函数

MATLAB 6.5 优化工具箱中用于求解方程解的函数有两个：fsolve 和 fzero，用法介绍如下。

### 2.8.1.1　fsolve 函数

**【功能】**

求解非线性方程（计算零点）。

**【工程应用背景】**

已知一个系统，它可以用一个非线性方程来表示，求解其零点。

**【语法】**

```
x = fsolve (fun, x0)
x = fsolve (fun, x0, options)
```

```
x = fsolve (fun, x0, options, P1, P2, ...)
[ x, fval ] = fsolve (...)
[ x, fval, exitflag ] = fsolve (...)
[ x, fval, exitflag, output ] = fsolve (...)
[ x, fval, exitflag, output, jacobian ] = fsolve (...)
```

**【说明】**

fun：是目标函数；

options：设置优化选项参数；

fval：返回目标函数在最优解 $x$ 点的函数值；

exitflag：返回算法的终止标志；

output：返回优化算法信息的一个数据结构。

**【实例分析】**

求下面两个方程所表示的系统的零点：

$$\begin{cases} 2x_1 - x_2 = e^{-x_1} \\ -x_1 + 2x_2 = e^{-x_2} \end{cases}$$

分析：

由题意，也即求解：

$$\begin{cases} 2x_1 - x_2 - e^{-x_1} = 0 \\ -x_1 + 2x_2 - e^{-x_2} = 0 \end{cases}$$ 的零点 $x$，取 $x0$=[-5, -5]

例程 2-14 是求解的 MATLAB 代码。

例程 2-14

```
%首先编写函数 f(x)
function F = myfun(x)
F = [2*x(1)-x(2)-exp(-x(1));   -x(1)+2*x(2)-exp(-x(2))];

x0 = [-5; -5]; %起始点
options = optimset('Display', 'iter'); %设置结果显示形式
[x, fval] = fsolve(@myfun, x0, options);
```

**【结果输出】**

```
x =
      0.5671
      0.5671
fval =
      1.0e-008
      -0.5320
      -0.5320
```

#### 2.8.1.2 fzero 函数

【功能】

求解单变量连续函数的零点。

【工程应用背景】

已知连续函数 $f(x)$，求 $f(x)=0$ 的根。

【语法】

```
x = fzero (fun, x0)
x = fzero (fun, x0, options)
x = fzero (fun, x0, options, P1, P2, ...)
[ x, fval ] = fzero (...)
[ x, fval, exitflag ] = fzero (...)
[ x, fval, exitflag, output] = fzero (...)
```

【说明】

fun：是目标函数；

options：设置优化选项参数；

fval：返回目标函数在最优解 $x$ 点的函数值；

exitflag：返回算法的终止标志；

output：返回优化算法信息的一个数据结构.

【实例分析】

求解函数 $f(x)=x^3-2x-5$ 在 $x$=2 附近的零点。

例程 2-15 是求解的 MATLAB 代码。

例程 2-15

```
%编写函数 f(x)
function y = f(x)
f = x.^3-2*x-5
z = fzero(@f, 2);
```

【结果输出】

```
z =
      2.0946
```

### 2.8.2 optimget 函数

【功能】

获取优化算法的特征参数值。

【工程应用背景】

优化函数的语法中都带有一个参数 options，它是一个结构参数，可以通过调用 optimget



函数来获取指定的 options 结构成员参数的值。

【语法】

```
val = optimget (options, 'param');
val = optimget (options, 'param', default)。
```

【说明】

一般来说，调用 optimget 函数时指定的 param 参数应该在 options 中有定义；如果 param 参数在 options 中没有定义，则返回 default 指定的参数值。

【实例分析】

要返回某一优化函数的 options 的 Display 的值。

分析：

```
val = optimget (options, 'Display')
```

### 2.8.3　optimset 函数

【功能】

设置或编辑优化算法的特征参数值。

【工程应用背景】

在实际应用中，可能需要对优化算法的特征结构参数 options 的某些成员参数进行具体设置或编辑以达到预期的效果，这时可以应用 optimset 函数。

【语法】

```
options = optimset ('param1', value1, 'param2', value2, …)
options = optimset (optimfun)
options = optimset (oldoptions, 'param1', value1, …)
options = optimset (oldoptions, newoptions)
```

【说明】

param 是 options 的成员特征参数，value 是其所要设置的值；oldoptions 代表以前的优化参数，newoptions 代表另外一个优化参数，从而用 newoptions 的非空参数值代替 oldoptions 中相应的参数的值。

【实例分析】

（1）设置 options 的 Display 属性为 'iter'，TolFun 的属性值为 1e-8：

```
options = optimset ('Display', 'iter', 'TolFun', 1e-8)
```

（2）将 oldoptions 的值复制给 newoptions：

```
newoptions = optimset (oldoptions,)
```

（3）将 oldoptions 的值复制给 newoptions，同时改变其中的 'TolX' 属性值为 1e-8：

```
newoptions = optimset (oldoptions, 'TolX', 1e-8)
```

# 2.9 综合范例演示

本节综合应用前面的知识，解决工程中的几个实际问题。通过这部分内容的学习，相信会对最优化问题的处理有更娴熟的掌握。

## 2.9.1 求解“香蕉”（Banana）函数的最小值

Banana 函数表达式为：$f(x)=100*(x(2)-x(1)^2)^2+(1-x(1))^2$

**【分析】**

Banana 函数的三维网格图如图 2-2（a）所示，Banana 函数的等高线图如图 2-2（b）所示。程序代码如例程 2-16 所示。

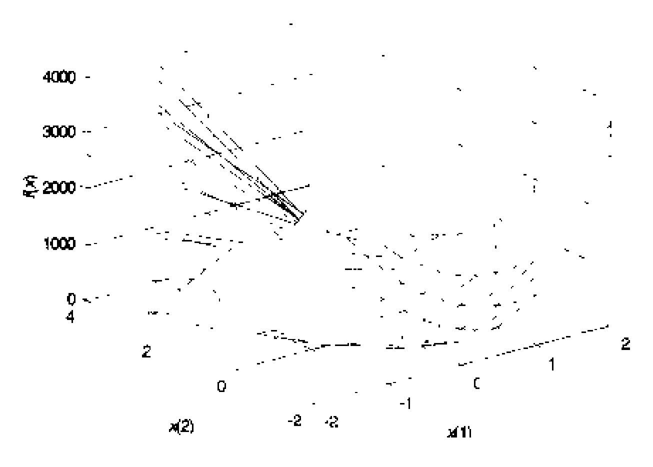



图 2-2（a） Banana 函数的三维网格图

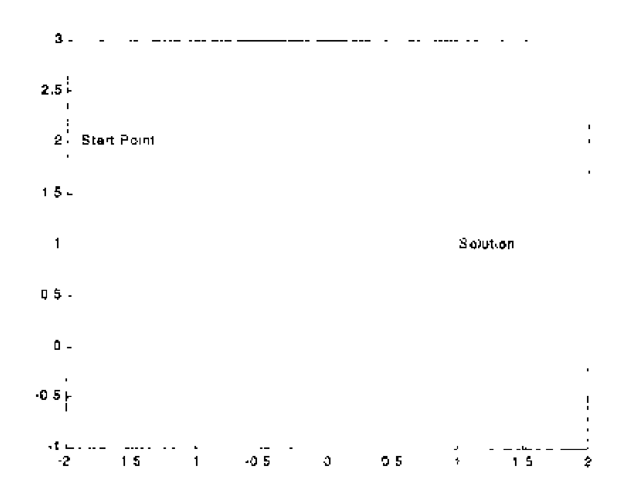



图 2-2（b） Banana 函数的等高线图

例程 2-16

```
Figure(1);
z1 = [-2:0.125:2]';
x2 = [-1:0.125:3]';
[x1,x2]=meshgrid(x1',x2') ;
meshd = 100.*(x2-x1.*x1).^2 + (1-x1).^2;
mesh(meshd)
xlabel('x(1)').
ylabel('x(2)');
zlabel('f(x)');

figure(2)
conts = exp(3:20);
xlabel('x1'),ylabel('x2'),title('Minimization of the Banana function')
contour(xx,yy,meshd,conts,'k:')
```

在本范例中，将使用优化工具箱中两种不同的标准算法来求解最小值，初始点为：$x=[-1.9，2]$。



函数 $f(x)$ 称为“香蕉”函数，是因为它的曲率绕原点变化形成的曲线形状像“香蕉”，也称做 Rosenbrock 函数。这个函数的特别之处在于采用大多数方法求解它的最小值时，收敛速度慢，它在[−1，1]之间存在惟一的最小值点：$f(x)=0$。

**【求解】**

[**思路一**]：当做一个无约束优化问题求解。

无约束优化问题的求解算法有多种形式，在优化工具箱中，可以通过设置不同的 options 参数来实现不同的优化算法。

1．BFGS（Broyden-Fletcher-Golfarb-Shanno）方法

例程 2-17 是求解的 MATLAB 代码。

例程 2-17

```
options=optimset('LargeScale','off');
options=optimset(options,'MaxFunEvals', 300);

%options=optimset(options,'LineSearch','cubicpoly');
% 采用本行参数设置时，表示计算方法为三次样条内插搜索法
% 否则为混合多项式内插搜索法

%以下部分在思路一的各算法程序中相同，后面程序中不再重复
x0=[-1.9,2]
GRAD='[100*(4*x(1)^3-4*x(1)*x(2))+2*x(1)-2: ...
100*(2*x(2)-2*x(1)^2); banplot2(x)]';
f='100*(x(2)-x(1)^2)^2+(1-x(1))^2';
```

```
options = optimset(options,'gradobj','on');
disp('[x,fval,exitflag,output] = fminunc({f,GRAD},x,options);');
[x,fval,exitflag,output] = fminunc({f,GRAD},x0,options);
```

【结果输出】

```
fval =
        8.98567e-009   %采用混合多项式内插搜索法
                       %图 2-3（a）表示其搜索过程
fval =
        2.2457e-010    %采用三次样条内插搜索法
                       %图 2-3（b）表示其搜索过程
```

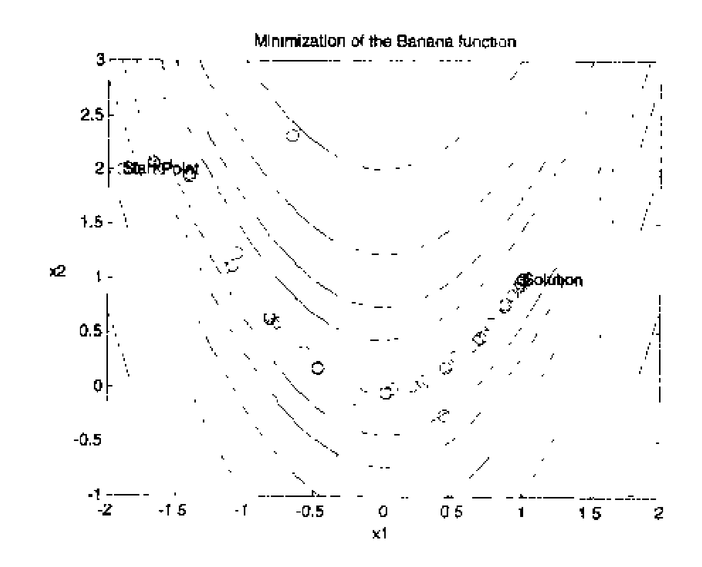



图 2-3（a） BFGS（Broyden-Fletcher-Golfarb-Shanno）方法 1

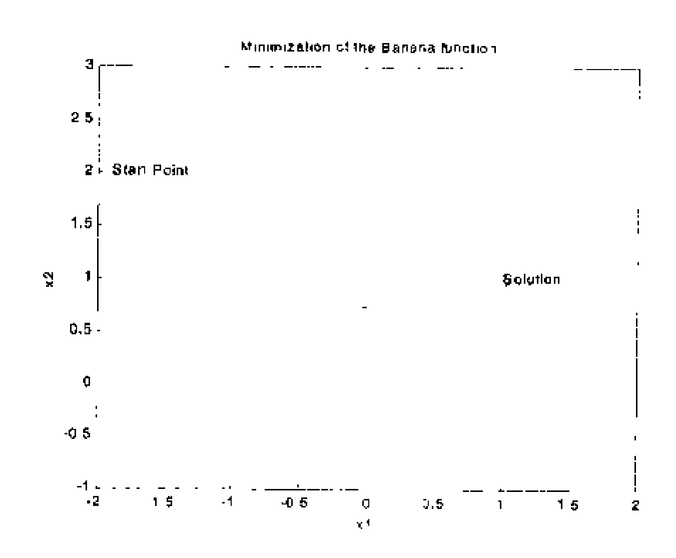



图 2-3（b） BFGS （Broyden-Fletcher-Golfarb-Shanno）方法 2

2、DFP（Davidon-Fletcher-Powell）方法

例程 2-18 是求解的 MATLAB 代码。

例程 2-18

```
options=optimset('LargeScale','off');
options=optimset(options,'MaxFunEvals', 300);
options=optimset(options,'HessUpdate','dft');
%options=optimset(options,'LineSearch','cubicpoly');
% 采用本行参数设置时，表示计算方法为三次样条内插搜索法
% 否则为混合多项式内插搜索法
%以下代码同上
...
```

**【结果输出】**

```
fval =
    0.0197909     %采用混合多项式内插搜索法
                  %图 2-4（a）表示其搜索过程
fval =
   2.2457e-010    %采用三次样条内插搜索法
                  %图 2-4（b）表示其搜索过程
```

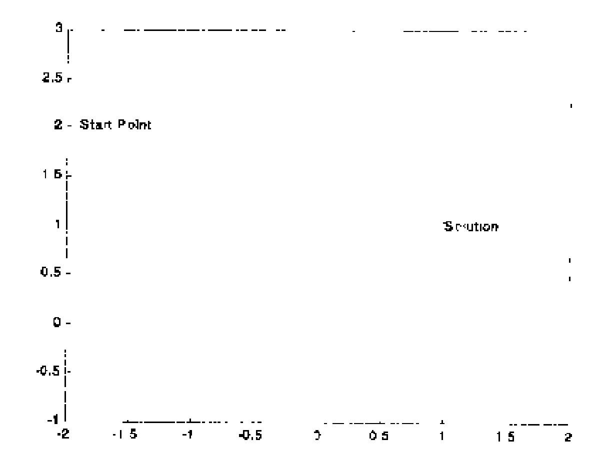



图 2-4 （a） DFP（Davidon-Fletcher-Powell）方法 1

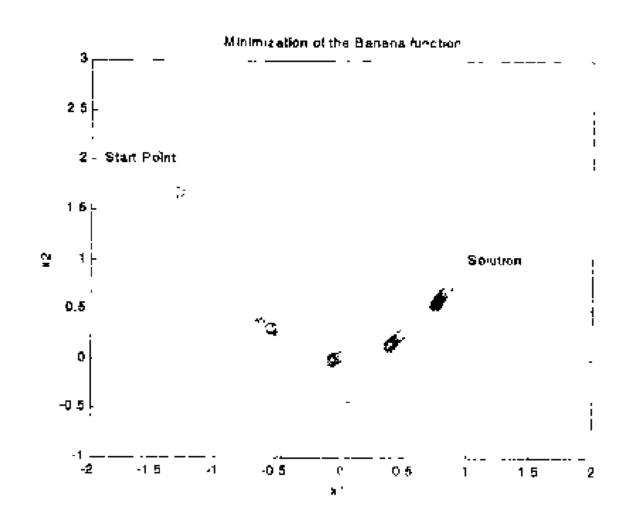



图 2-4（b） DFP（Davidon-Fletcher-Powell）方法 2

3．最速下降法

例程 2-19 是求解的 MATLAB 代码。

例程 2-19

```
options=optimset('LargeScale','off');
options=optimset(options,'MaxFunEvals', 300);

options=optimset(options,'HessUpdate','steepdesc');
%options=optimset(options,'LineSearch','cubicpoly');
% 采用本行参数设置时，表示计算方法为三次样条内插搜索法
% 否则为混合多项式内插搜索法
% 以下代码同上
...
```

【结果输出】

```
fval =
        4.84404      %采用混合多项式内插搜索法
                     %图 2-5（a）表示其搜索过程
fval =
        4.81378      %采用三次样条内插搜索法
                     %图 2-5（b）表示其搜索过程
```

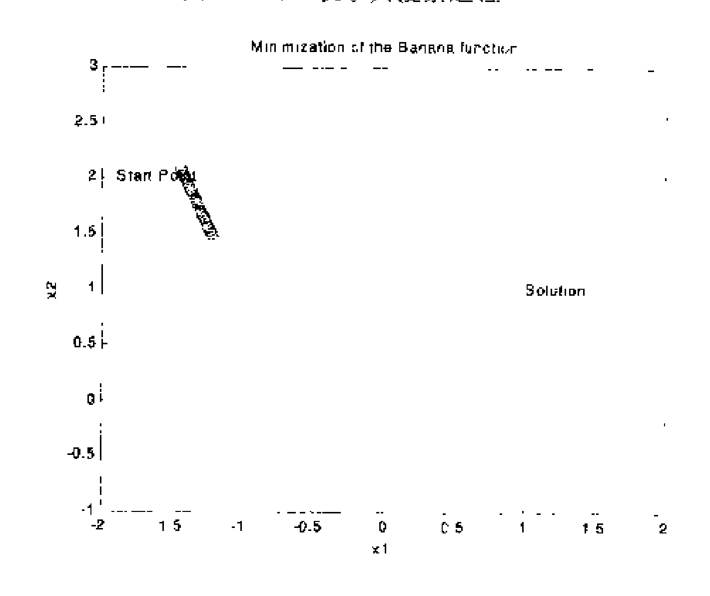



图 2-5（a）　最速下降法 1




最速下降法仅是线性收敛的，并且有时是很慢的线性收敛，因此在算法实现过程中受估计函数值次数的限制，可能得不到收敛解。这表明最速下降法对一些病态的优化问题是不适合的。

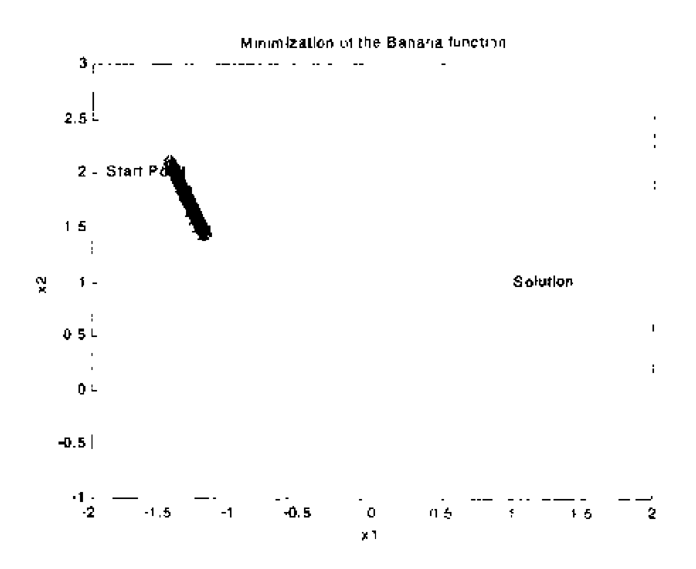



图 2-5（b）　最速下降法 2

4．单纯搜索法

例程 2-20 是求解的 MATLAB 代码。

例程 2-20

```
options=optimset('LargeScale','off');
options=optimset(options,'MaxFunEvals', 300);

x0=[-1.9,2];
f='[100*(x(2)-x(1)^2)^2+(1-x(1))^2; banplot2(x)]';
disp('[x,fval,exitflag,output] = fminsearch(f,x0, options);');
[x,fval,exitflag,output] = fminsearch(f,x0, options);
```

【结果输出】

```
fval =
4.06855e-010
% 搜索过程如图 2-6 所示
```

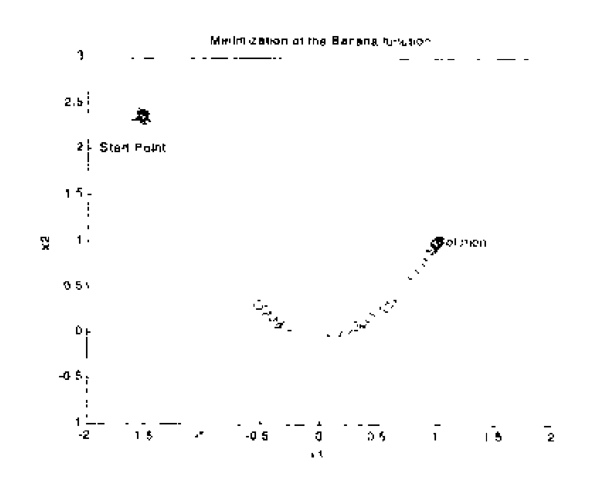



图 2-6　单纯搜索法

[思路二]：当做一个最小二乘优化问题。

1．Gauss-Newton 方法

例程 2-21 是求解的 MATLAB 代码。

例程 2-21

```
options=optimset('LargeScale','off');
options=optimset(options,'MaxFunEvals', 300);
options=optimset(options,'LevenbergMarquardt','off');

%options=optimset(options,'LineSearch','cubicpoly');
% 采用本行参数设置时，表示计算方法为三次样条内插搜索法
% 否则为混合多项式内插搜索法
x0=[-1.9,2];
JAC='[-20*x(1), 10; -1, 0; banplot2(x)]';
f='[10*(x(2)-x(1)^2),(1-x(1))]';
options = optimset (options,'Jacobian','on');
disp('[x,resnorm,residual,exitflag,output]=lsqnonlin({f,JAC},x0,[],[], options);');
[x,resnorm,residual,exitflag,output]=lsqnonlin({f,JAC},x0,[],[], options);
fval = resnorm;
```

【结果输出】

```
fval =
        1.24492e-030    %采用混合多项式内插搜索法
                        %图 2-7（a）表示其搜索过程
fval =
        3.67793e-021    %采用三次样条内插搜索法
                        %图 2-7（b）表示其搜索过程
```

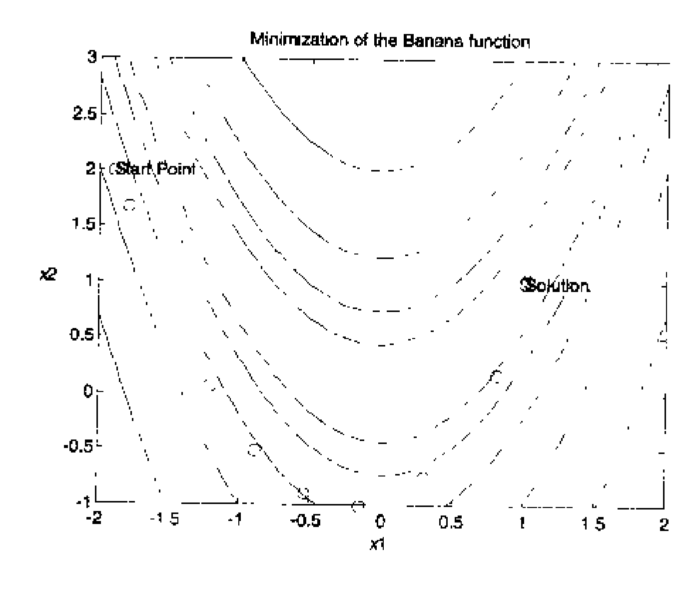



图 2-7（a） Gauss-Newton 方法 1

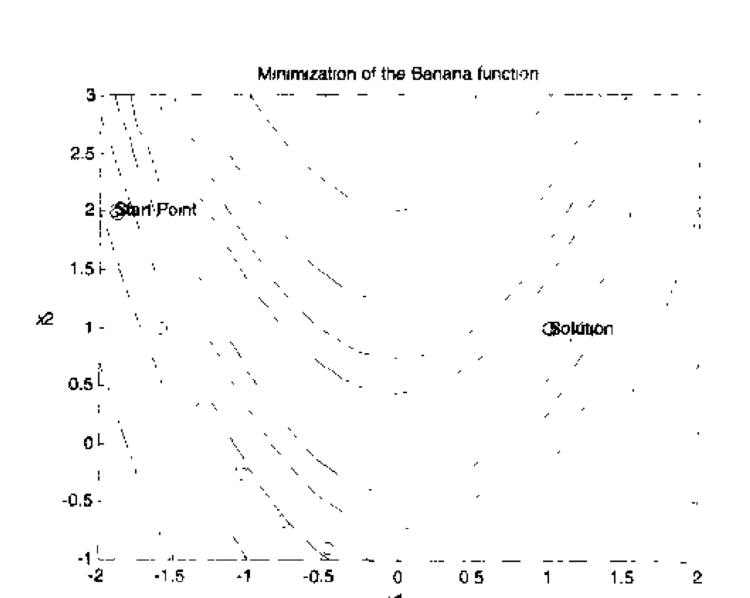



图 2-7（b） Gauss-Newton 方法 2

2．Levenberg-Marquardt 方法

例程 2-22 是求解的 MATLAB 代码。

例程 2-22

```
options=optimset('LargeScale','off');
options=optimset(options,'MaxFunEvals', 300);

options=optimset(options,'LineSearch','cubicpoly');
%用本行参数设置时，表示计算方法为三次样条内插搜索法
%否则为混合多项式内插搜索法
x0=[-1.9,2];
JAC='[-20*x(1), 10; -1, 0; banplot2(x)]';
f='[10*(x(2)-x(1)^2),(1-x(1))]';
options = optimset (options,'Jacobian','on');
disp('[x,resnorm,residual,exitflag,output]=lsqnonlin({f,JAC},x0,[],[], options);');
[x,resnorm,residual,exitflag,output]=lsqnonlin({f,JAC},x0,[],[], options);
fval = resnorm;
```

**【结果输出】**

```
fval =
            1.02672e-020    %采用混合多项式内插搜索法
                            %图 2-8（a）表示其搜索过程
fval =
            5.88833e-014    %采用三次样条内插搜索法
                            %图 2-8（b）表示其搜索过程
```

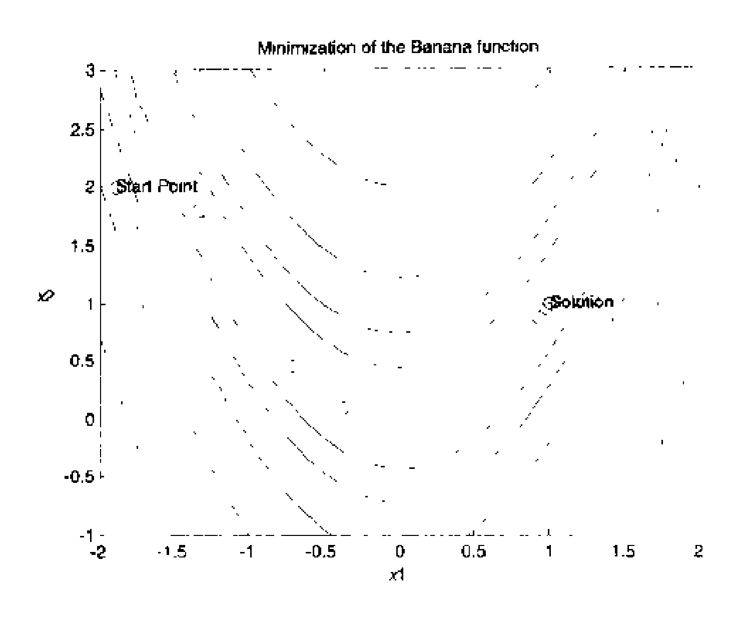



图 2-8（a） Levenberg-Marquardt 方法 1

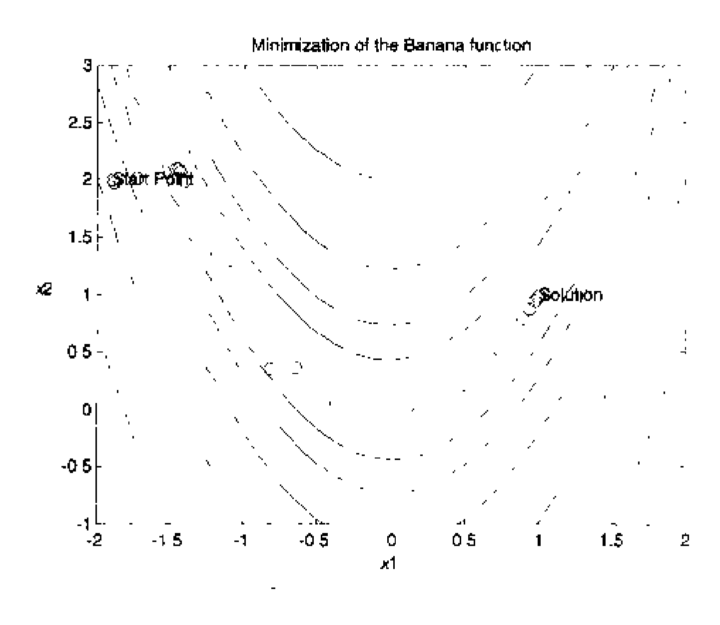



图 2-8（b） Levenberg-Marquardt 方法 2

### 2.9.2 不稳定系统的求解

已知一个 2 输入-2 输出不稳定系统如图 2-9 所示，状态空间方程为：

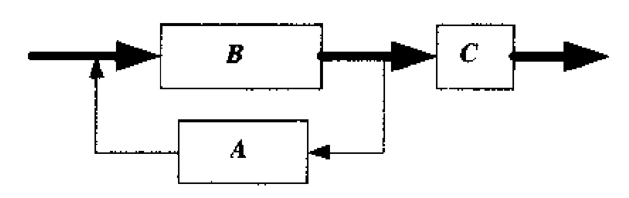



图 2-9 2 输入-2 输出不稳定系统



$$\begin{cases} \dot{X} = A * X + b \\ y = C * x \end{cases}$$

其中：

$$A = \begin{bmatrix} -0.5 & 0 & 0 \\ 0 & -2 & 1 \\ 0 & 10 & -2 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 0 \\ -2 & 2 \\ 0 & 1 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

要求设计一个输出反馈器，使得系统的极点位于复平面上的[-5，-3，-1]，且反馈器的增益不大于 4，从而系统成为一个稳定系统。

【分析】

这个问题可以转化为一个目标逼近问题来求解，目标 $GOAL = [-5, -3, -1]$。

【求解】

例程 2-23 是求解的 MATLAB 代码。

例程 2-23

```
clc
echo on
A =  [ -0.5  0  0;  0  -2  10;  0  1  -2 ];
B =  [ 1  0;  -2  2;  0  1 ];
C =  [ 1  0  0;  0  0  1 ];

GOAL = [-5, -3, -1];
WEIGHT = abs(GOAL)

% 初始化输出反馈器的参数值
X0 = [ -1 -1; -1 -1];
UB =   4*ones(2,2)
LB = -4*ones(2,2)
OPTIONS = [];           %取默认值
OPTIONS = optimset ('Display','iter');
[x,fval, attainfactor,exitflag,output,lambda] = fgoalattain('eigfun',x0,GOAL,WEIGHT,[],[],[],[],...
LB,UB,[],OPTIONS,A,B,C);
%显示优化得到的反馈控制器参数
x
% 显示闭环系统的特征向量
OPTIONS = optimset (OPTIONS,'GoalsExactAchieve',3);
 [X, fval, attainfactor, exitflag, output, lambda] = fgoalattain('eigfun',X0,GOAL,WEIGHT,[],[],
[],[],LB,...
UB,[],OPTIONS,A,B,C);
eigfun(X,A,B,C)
```

【结果输出】

```
UB =
        4       4
        4       4
LB =
       -4      -4
       -4      -4
```

迭代过程参数如表 2-1 所示。

表 2-1 迭代参数

| Iter | F-count | Attainment factor | Step-size | Directional derivative | Procedure |
|---|---|---|---|---|---|
| 1 | 6 | 1.885 | 1 | 1.03 | |
| 2 | 13 | 1.061 | 1 | -0.679 | |
| 3 | 20 | 0.4211 | 1 | -0.523 | Hessian modified |
| 4 | 27 | -0.06352 | 1 | -0.053 | Hessian modified twice |
| 5 | 34 | -0.1571 | 1 | -0.133 | |
| 6 | 41 | -0.3489 | 1 | -0.00768 | Hessian modified |
| 7 | 48 | -0.3643 | 1 | -4.25e-005 | Hessian odified |
| 8 | 55 | -0.3645 | 1 | -0.00303 | Hessian modified twice |
| 9 | 62 | -0.3674 | 1 | -0.0213 | Hessian modified |
| 10 | 69 | -0.3806 | 1 | 0.00266 | |
| 11 | 76 | -0.3862 | 1 | -2.73e-005 | Hessian modified twice |
| 12 | 83 | -0.3863 | 1 | -1.2e-013 | Hessian modified twice |

```
x =
     -4.0000    -0.2564
     -4.0000    -4.0000
优化后极点值
ans =
      -6.9313
      -4.1588
      -1.4099
```

## 2.9.3 曲线拟合问题

已知 $x_{data}$ ={0,0.1000,0.2000,0.3000,0.4000,0.5000,0.6000,0.7000,0.8000,0.9000,1.000,1.1000,1.2000,1.3000,1.4000,1.5000,1.6000,1.7000,1.800,1.9000,2.000}；

$y_{data}$ ={ 5.8955,3.5639,2.5173,1.9799,1.8990,1.3938,1.1359,1.0096,1.0343,0.8435,0.6856,0.6100,0.5392,0.3946,0.3903,0.5474,0.3459,0.1370,0.2211,0.1704,0.2636}。要求按下式拟合曲线：



$$y = c(1) * e^{-d(1)*x} + c(2) * e^{-d(2)*x}$$

其中：$c(1)$，$c(2)$为两个线性参数；

$d(1)$，$d(2)$为两个非线性参数。

**【分析】**

（$x_{data}$，$y_{data}$）在直角坐标系中的图形如图 2-10 所示。

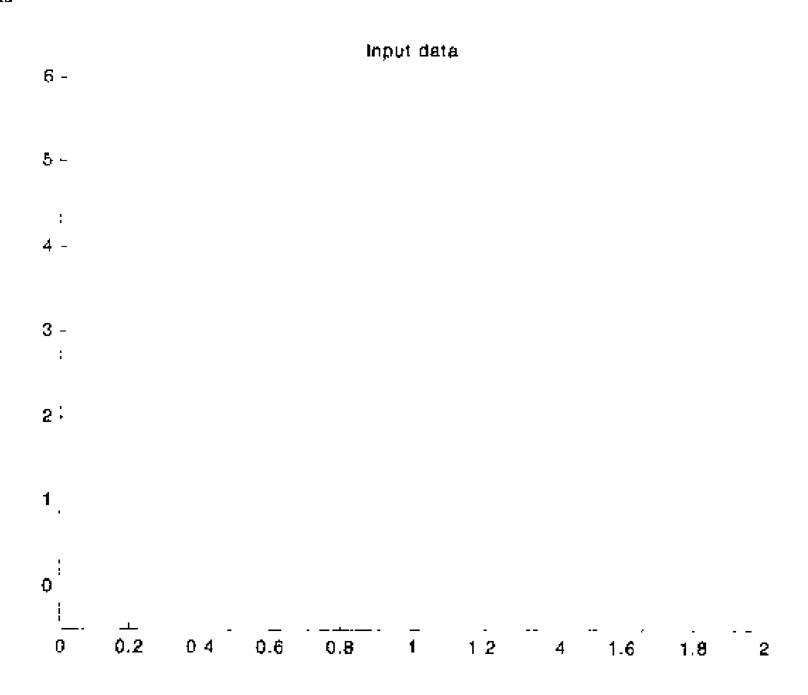



图 2-10　（$x_{data}$，$y_{data}$）在直角坐标系中的图形

因为拟合的目标函数中既含有线性参数，又含有非线性参数，因此在求解过程中将分两步走：首先用函数 lsqnonlin 解决非线性参数，然后用 "\" 解决线性参数。

（1）编写.m 函数 myfun，输入参数为给定的拟合数据和待定的非线性参数 $d$；输出为线性参数 $c$ 的估计值和拟合误差；

（2）调用优化函数。

**【求解】**

例程 2-24 是求解的 MATLAB 代码。

例程 2-24

```
function f =myfun(d,Data)
%假设 y =c(1)*exp(-d(1)*x) + … + c(n)*exp(-d(n)*x)
%带有 n 个线性参数，n 个非线性参数
x = Data(:,1); y = Data(:,2); % 获得 x 和 y 的值
A = zeros(length(x),length(d));   % 初始化矩阵 A
for j = 1:length(d)
    A(:,j) = exp(-d(j)*x);
```

```
end
c = A\y;        %通过 A*c = y 求解线性参数 c
z = A*c;
f = z-y;

Data=[0,0.1000,0.2000,0.3000,0.4000,0.5000,0.6000,0.7000,
8000,0.9000, .000,1.1000,1.2000,1.3000,1.4000,1.5000,
1.6000,1.7000,1.8000,1.9000,2.000;
5.8955,3.5639,2.5173,1.9799,1.8990,1.3938,1.1359,1.0096,
1.0343,0.8435,0.6856,0.6100,0.5392,0.3946,0.3903,0.5474,
0.3459,0.1370,0.2211,0.1704,0.2636]';
options=optimset('LargeScale','off');
options=optimset(options,'MaxFunEvals', 300);
options=optimset(options,'LevenbergMarquardt','off');
options=optimset(options,'LineSearch','cubicpoly');

d0=[1,2];
d =lsqnonlin('myfun2',d0,[],[], options, Data);
```

**【结果输出】**

```
c =
        2.8898
        3.0061
d =
        1.4006    10.5891
```

即

$$y = 2.8898 * e^{-1.4006*x} + 3.0061 * e^{-10.5891*x}$$

拟合曲线如图 2-11 所示。

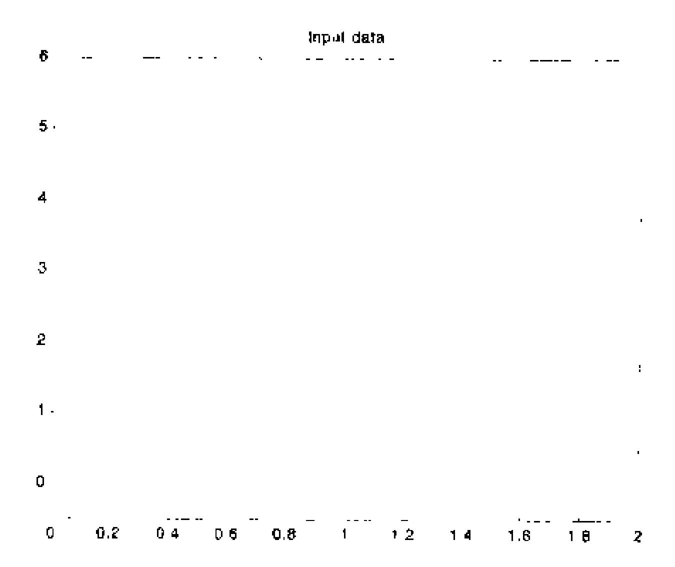



图 2-11 拟合曲线

## 2.10　优化工具箱函数使用的常见问题及对策

求解优化问题常常需要经过多步迭代，最终收敛到最优解，因此最优解对诸如计算有限差分梯度时的截断误差之类的数字问题比较敏感。大多数优化问题的解决很大程度上是因为取了一个较好的初始点，这样的初始点提高了优化算法的执行效率，而且能够求得优化问题的全局最优解，而不是局部最优解。在优化问题求解的初始阶段，设置简单的优化效用函数和松弛的算法终止条件往往可以减少计算时间，甚至可以避免局部最优解而得到更好的优化结果。

MATLAB 6.5 的优化工具箱可以解决很多类型的优化问题，许多优化技术遇到的限制在这里可以通过一定的转换来加以克服，以下就是一些典型的常见问题及解决对策。

**【问题一】**

在求解最优化问题中，似乎得不到全局最小值。

**【解决对策】**

一般来说，不能保证待求的最优化问题有全局最小值，除非问题本身是连续的且只有一个最小值。但是在求解时，我们可以取多组不同的初始点进行计算，这种方法有助于得到全局最小值或证实待求的最优化问题有惟一的全局最小值。另外，使用不同的优化算法也有助于改进优化结果。

**【问题二】**

fminunc 提示警告信息：在接近最优解时，收敛速度慢。

**【解决对策】**

这是因为用户在使用 fminunc 函数时，使用的不是经过系统分析得到的梯度，而且算法终止条件很严格。因此，受到以差分方式计算梯度时的截断误差的影响，优化算法在接近最优解时收敛速度变慢。解决这一问题的方法是放松算法终止条件，另外对于标准算法而言，还可以改变参数 DiffMinChange 和 DiffMaxChange 的值，以提高计算梯度的准确性。

**【问题三】**

待求的优化问题是一个不连续函数。

**【解决对策】**

很多优化算法的实现都是以待优化函数的一阶或二阶导数为基础的，如果待优化函数的不连续点不在最优解附近，用常用的优化算法可能会得到最优解。解决这一问题的方法是平滑不连续的待优化函数，比如对目标函数进行插值平滑；另外对于标准算法，可以修改参数 DiffMinChange 和 DiffMaxChange 的值，从而跳过不连续点。

**【问题四】**

有时候，优化问题目标函数或非线性约束函数中的优化变量 $x$ 对某个点没有定义，也就是说在某个点处，无法计算目标函数或非线性约束函数的值。

【解决对策】

对于这一类问题，可以为变量加一个边界约束条件以避开没有定义的点。另外，还可以加一个惩罚函数，当搜索点接近该点时，使得目标函数或非线性约束函数的值变得很大，使得搜索点不回落到该点处，对利用梯度信息的算法而言，惩罚函数应该是光滑连续的。

【问题五】

待优化的独立变量 $x$ 的值必须是离散的值，比如说整数。

【解决对策】

这类问题常常发生在变量 $x$ 所代表的实际意义必须符合某种标准的计量单位时，比如物体的数目等。虽然 MATLAB 6.5 的优化工具箱没有明确表明能够解决离散的优化问题，但是它可以通过求解一个等价的连续问题来解决，在求解过程中首先对第一个变量的值进行上下取整运算得到最近且最优的离散值，从而解决一个变量的离散化问题。依次类推，当所有的变量都取到最优的离散值后，这类问题也就得到解决了。

【问题六】

最小化的搜索过程似乎陷入了一个死循环或者得到的最优解不符合约束条件的限制。

【解决对策】

发生这类问题可能是因为在求解过程中待求的目标函数、约束函数或者梯度函数的值出现了 NaN、INF 或更复杂的结果，而最小化过程需要目标函数、约束函数或者梯度函数的值为实数。解决这一问题可以加入校验函数来检查这些函数的返回值是否为实数，从而保证目标函数、约束函数或者梯度函数的返回值为实数，比如 isfinite 函数。

【问题七】

使用 lsqnonlin 函数得不到期望的收敛结果。

【解决对策】

避免这一问题的方法就是将目标函数明确写成平方和的标准形式，因为 lsqnonlin 在实现时需要向量或矩阵中的函数值是一个平方和。

【问题八】

如何求解极大值问题？

【解决对策】

在 MATLAB 6.5 优化工具箱中，优化函数 fminbnd、fminsearch、fminunc、fmincon、fgoalattain、fminmax、lsqcurvefit、lsqnonlin 都是求解目标函数 $f(x)$的极小值。对极大值问题，可以通过求$-f(x)$的极小值来实现。

# 第3章 工程优化算法及其MATLAB实现（一）——标准算法

本章通过大量的范例全面介绍求解各种常见优化问题的标准算法的 MATLAB 实现，帮助读者熟练使用 MATLAB 优化工具箱来解决工程实际问题。

本章主要内容包括：

- 无约束优化算法及实现
- 约束优化算法及实现
- 最小二乘优化算法及实现
- 多目标优化算法及实现

## 3.1 引　言

优化技术常用来确定一组设计参数 $\boldsymbol{x}=\{x_1, x_2, \cdots, x_n\}$，使得这些参数在一定的评价准则下是最优的。对于最简单的情况，比如需要计算系统中某个与参数 $\boldsymbol{x}$ 有关的特征量的最小或最大值。更复杂的情况就是求出目标函数 $f(\boldsymbol{x})$的最小值或最大值，其中 $\boldsymbol{x}$ 满足一定的等式 $G_i(\boldsymbol{x})=0$（$i=1,\cdots, j$）或不等式 $G_i(\boldsymbol{x})\leqslant 0$ （$i=j+1, \cdots, n$）约束条件，甚至有可能参数 $x$ 本身有一定的区间大小限制（$\boldsymbol{l}\leqslant \boldsymbol{x}\leqslant \boldsymbol{ub}$）。

优化问题的最一般形式可以描述如下：

$$\min_{x\in \Re^{\Omega}} f(\boldsymbol{x})\text{，且 } \boldsymbol{x} \text{ 满足约束条件}$$

$$\begin{cases} G_i(\boldsymbol{x})=0 & i=1,\cdots,j \\ G_i(\boldsymbol{x})\leqslant 0 & i=j+1,\cdots,n \\ \boldsymbol{lb}\leqslant \boldsymbol{x}\leqslant \boldsymbol{ub} \end{cases} \qquad \text{（式 3.1）}$$

其中，

$\boldsymbol{x}$：需要确定的参数向量（$\boldsymbol{x}\in R^{\Omega}$）；

$f(\boldsymbol{x})$：指定的目标函数，返回值为标量，即 $f(x)$：$R^{\Omega}\to R$；

$G(\boldsymbol{x})$：返回在 $\boldsymbol{x}$ 点处的等式和不等式约束的值，返回值为向量，即 $G(\boldsymbol{x})$：$R^{\Omega}\to R^{\Omega}$。

一般来说，能否得到最优化问题的正确解及求解过程的效率是否高不仅取决于约束因素和优化变量的数目多少，还取决于目标函数和约束条件的性质。当目标函数和约束条件均是优化变量的线性函数时，我们称这一类优化问题为线性规划（Linear Programming）问题；如果目标函数是一个优化变量的二次型函数，且约束条件为优化变量的线性函数，

我们称这一类问题为二次规划（Quadratic Programming）问题；如果目标函数或约束条件是优化变量的非线性函数，我们称这一类问题为非线性规划（Nonlinear Programming）问题。非线性规划问题的求解往往比较难，通常需要用一定的手段将其转化为线性规划或二次规划形式的子问题来解决



本章所指的“标准算法”并不是一个专门的数学术语，而是相对第 4 章的“大规模算法”给出的一个概念名词，以表示这两种算法的不同。

# 3.2 无约束优化算法及实现

无约束优化问题的基本形式如下：

$$\min_{x} f(x), \qquad x \in R^n$$

目前，在实际应用中求解无约束优化问题的算法很多，但是根据其在实现过程中是否使用了函数的导数信息这一准则，我们可以将无约束优化算法分为广义上的两大类：搜索法和梯度法。

搜索法是解决非线性或不连续问题的一种有效的方法，而梯度法使用目标函数的导数信息来确定每一次搜索的方向，特别当欲优化的目标函数具有连续的一阶导数时，这种方法更显得有效。另外，如果被优化的目标函数的二阶导数信息能够很方便地求出来，我们还可以采用高阶法来求解它，比如牛顿法。（当然，如果二阶导数信息不能很方便地求出来，那么不能用高阶法，因为计算机实现这一算法时用差分的形式计算导数会带来很大的计算开销）使用梯度法的一个简单的例子就是沿$-\nabla f(x)$方向进行搜索，其中$\nabla f(x)$是目标函数的梯度，不过当被优化的目标函数 $f(x)$具有窄长形的谷值时，这一方法的效率很低，甚至不能得到优化解。

## 3.2.1 拟牛顿（Quasi-Newton）方法

### 3.2.1.1 算法描述

在所有实现过程里使用了梯度信息的优化算法中，拟牛顿（Quasi-Newton）方法是最常用的一个，此方法的实质是每一次迭代时建立其曲率信息，并以此来解决如下形式的二次模型优化问题：

$$\min_{x \in R^n} f(x) = \min_{x \in R^n} \left\{ \frac{1}{2} x^T \boldsymbol{H} x + \boldsymbol{b}^T x + c \right\}, \qquad \text{（式 3.2）}$$

其中：$\boldsymbol{H}$ 为正定对称矩阵（Hessian），$\boldsymbol{b}$ 为常数向量，$c$ 为常数。此问题的最优解在对 $x$ 的偏导数为零的点处，即：

$$\nabla f(x^*) = \boldsymbol{H} x^* + \boldsymbol{b} = 0$$

从而，最优解可以写为：　　$x^* = -H^{-1}b$

牛顿法与拟牛顿法相对，它直接计算矩阵 $H$，并使用线性搜索法沿着下降方向搜索，经过一定的迭代次数后确定最小值，在此过程中，为了求得矩阵 $H$，算法要经过大量的计算；而拟牛顿法则不同，它通过观测 $f(x)$和$\nabla f(x)$的行为计算曲率信息，然后经过恰当的转换来估计 $H$ 的近似值。

考虑如（式 3.2）所示的优化问题，

**1．采用牛顿法**

设 $f(x)\in C^2$ 的一般函数，则在 $x^{(k)}$的局部有：

$$f(x) \approx q_k(x) = f(x^{(k)}) + \nabla f(x^{(k)})^T (x - x^{(k)}) + \frac{1}{2}(x - x^{(k)})^T \nabla^2 f(x^{(k)})(x - x^{(k)})$$

记 $g_k = \nabla f(x^{(k)}), G_k = \nabla^2 f(x^{(k)})$，当 $\nabla^2 f(x^{(k)})$ 正定时，则 $q_k(x)$ 存在惟一极小点，

将它取为 $x^*$的下一个近似值 $x_{k+1}$，则应满足 $\nabla q_k(x_{k+1}) = 0$，即：

$$G_k(x_{k+1} - x_k) + g_k = 0$$

$$x_{k+1} = x_k - G_k^{-1} g_k$$

于是算法流程图如图 3-1 所示。

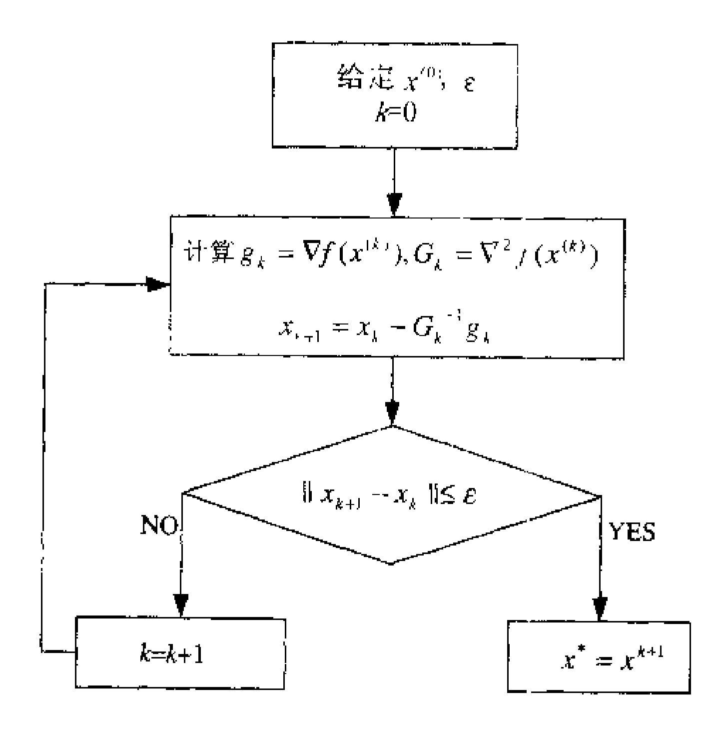



图 3-1　算法流程图

**2．采用拟牛顿法**

为了避免计算 $G_k = \nabla^2 f(x^{(k)})$ 及其逆矩阵的大计算量，于是又有一种设想，将牛顿

法中的$G_k^{-1}=\nabla^2 f(x^{(k)})^{-1}$用 $n$ 阶矩阵 $\boldsymbol{H}_k$ 代替。确定 $\boldsymbol{H}_k$ 的一种自然想法就是将 $\boldsymbol{H}_k$ 作为$G_k^{-1}$的近似来构造。又注意到$\nabla^2 f(x^{(k)})$是对称矩阵，且有近似关系式：

$$\nabla f(x^{k-1}) \approx \nabla f(x^{(k)})+\nabla^2 f(x^{(k)})(x^{(k-1)}-x^k)$$
$$(x^{(k)}-x^{(k-1)}) \approx \nabla^2 f(x^{(k)})^{-1}(\nabla f(x^{(k)})-\nabla f(x^{(k-1)}))$$

若记$\nabla f(x^{(k)})=g_k,\gamma_k=g_k-g_{k-1},\delta_k=x^{(k)}-x^{(k-1)}$，因此要求 $\boldsymbol{H}_k$满足：对称和拟牛顿方程$\boldsymbol{H}_k\gamma_k=\delta_k$。有很多 $\boldsymbol{H}$ 矩阵的估计方法，一般说来，Broyden、Fletcher、Goldfarb 和 Shanno（BFGS）等人的公式被认为是解决这类问题最为有效的方法，公式如（式 3.3）所示。

$$\boldsymbol{H}_{k+1}=\boldsymbol{H}_k+\frac{q_k q_k^T}{q_k^T s_k}-\frac{\boldsymbol{H}_k^T\boldsymbol{H}_k}{s_k^T\boldsymbol{H}_k s_k} \qquad \text{（式 3.3）}$$

其中：

$$s_k=x_{k+1}-x_k$$
$$q_k=\nabla f(x_{k+1})-\nabla f(x_k)$$

初值 $\boldsymbol{H}_0$ 可以是任意的对称正定矩阵，例如可取单位矩阵，为了避免对 Hessian 矩阵的求逆计算，可以推导一个在每次迭代时对 Hessian 矩阵的逆进行近似估计的更新方法。

#### 3.2.1.2 拟牛顿算法的MATLAB 实现

在MATLAB 优化工具箱中，函数 fminunc 使用了拟牛顿法，它的实现包括两个阶段：

- 确定搜索方向（即更新 Hessian 矩阵）；
- 线性搜索过程。

两个阶段的具体实现如下所示。

**1．Hessian 矩阵更新**

搜索方向是由选择 BFGS 方法还是选择 DFP 方法来决定的（在设置 options 参数时，将“HessUpdate”以“dfp”代替就可以确定 DFP 方法），因为 Hessian 矩阵 $\boldsymbol{H}$ 总是保持正定的，所以使得搜索方向 $d$ 总是在下降方向，这也意味着对于搜索方向 $d$ 上任意小的步长 $\alpha$，目标函数的值在不断减小。只要 $\boldsymbol{H}$ 的初始值为正定的，而且计算出来的 $q_k^T s_k$ 总是正定的，那么 $\boldsymbol{H}$ 的正定性就能够得到保证。$q_k^T s_k$ 可以由下式得到：

$$q_k^T s_k=\alpha_k(\nabla f(x_{k+1})^T d-\nabla f(x_k)^T d)$$

只要执行足够精度的线性搜索，$q_k^T s_k$ 为正的条件总能够得到满足，这是因为搜索方向 $d$ 是下降的，使得$\alpha_k$和$-\nabla f(x_k)^T$总为正，因此可能为负的项$\nabla f(x_k)^T d$随着线性搜索精

度的增大，其幅值能达到一个设定的小的值。

2．线性搜索过程

根据梯度信息是否容易得到还是必须通过有限微分方法来计算，我们可以将线性搜索方法分为两种：当梯度信息容易得到时，默认情况是使用三次多项式方法；当梯度信息不容易得到时，默认情况是使用混合二次和三次多项式方法。下面介绍三次多项式方法。

在所提出的三次多项式方法中，每一个迭代周期 $k$ 都要进行梯度和函数值的计算，即在每一次迭代中，当找到一个满足：

$$f(x_{k+1}) < f(x_k), \qquad \text{（式 3.4）}$$

的新的迭代点 $x_{k+1}$ 时，都要进行更新。

每一次迭代时，由迭代步长 $\alpha_k$ 得到下一次的迭代值，即：

$$x_{k+1} = x_k + \alpha_k * d$$

如果（式 3.4）不能满足时，就减小 $\alpha_k$ 得到下一个新的步长 $\alpha_{k+1}$，这常常使用均分的方法来实现（也即连续取上一个步长的一半直至满足条件为止）。不过这种方法同使用三次插值、梯度和函数估计值来推导迭代步长的方法相比，它的计算过程是较慢的。

### 3.2.2 线性搜索方法

大多数无约束求解算法通过求解一个子问题得到一个搜索方向，而优化问题的解可能位于这个搜索方向上。通常沿着这个搜索方向，使用一定的搜索过程（比如 Fibonacci、Gold Section）或多项式插值方法（比如二次、三次插值）来计算目标函数的最小值。多项式方法使用一个最小值容易计算的单变量多项式得到一系列的逼近点，一般来说，如果目标函数是连续函数，那么多项式插值方法就执行效率而言是最有效的。线性搜索的主要问题就是根据下式得到一个新的迭代点：

$$x_{k+1} = x_k + \alpha_k * d$$

其中，$x_k$ 表示当前的迭代值；$x_{k+1}$ 表示下一个迭代值；$d$ 表示搜索方向；$\alpha_k$ 表示步长的一个标量值。

多项式插值方法包括二次内插值和三次内插值。

1．二次内插值方法

二次内插值方法用于解决由数据来拟合如下形式的单变量函数的问题：

$$f(x) = a\alpha^2 + b\alpha + c$$

其中极值以步长得到 $\alpha^* = -\dfrac{b}{2a}$，此点的值可能是最小值或最大值，当执行内插或 $\alpha$ 为正时是最小值。利用任何三梯度组合或函数计算都能确定系数 $a$ 和 $b$，上面的计算也可以仅用两梯度得到，此时系数由建立的线性方程组的解得到。

二次内插值方法的一般形式是：在定义空间中，给定三个点$\{x_1,x_2,x_3\}$和它们对应的函数值（$f(x_1),f(x_2),f(x_3)$），由二阶拟合得出的最小解如下：

$$x_{k+1}=\frac{1}{2}\frac{\beta_{23}f(x_1)+\beta_{31}f(x_2)+\beta_{12}f(x_3)}{2\gamma_{23}f(x_1)+\gamma_{31}f(x_2)+\gamma_{12}f(x_3)}$$

其中：

$$\beta_{ij}=x_i^{\ 2}-x_j^{\ 2}$$

$$\gamma_{ij}=x_i-x_j,\{i,j\}=\{2,3\},\{3,1\},\{1,2\}$$

2. **三次内插值方法**

当梯度信息可以容易得到或者被优化函数的估计值数目超过 3 个时，三次内插值方法是很有用的。三次内插值方法常用于满足以下形式的单变量函数的数据拟合问题：

$$f(x)=a\alpha^3+b\alpha^2+c\alpha+d$$

其中局部极值是下面二次方程的根：

$$3a\alpha^{*2}+2b\alpha^*+c=0$$

为了求得最小极值，选择使$6a\alpha^{*2}+2b\alpha^*+c=0$为正的根，利用任何四梯度组合或函数估计都能确定系数 $a$ 和 $b$，另外，系数也可能仅用三梯度就可以得到，此时系数由一个联立的线性方程的解的公式决定。

三次内插值方法的一般形式是：在定义域空间中给定两个点$\{x_1, x_2\}$和被优化函数分别在这两点处的梯度值$\{\nabla f(x_1), \nabla f(x_2)\}$，以及相应的函数值$\{f(x_1), f(x_2)\}$，则最小解如下所示：

$$x_{k+1}=x_2-(x_2-x_1)\frac{\nabla f(x_2)+\beta_2-\beta_1}{\nabla f(x_2)-\nabla f(x_1)+2\beta_2}$$

其中：

$$\beta_1=\nabla f(x_1)+\nabla f(x_2)-3\frac{f(x_1)-f(x_2))}{x_1-x_2}$$

$$\beta_2=[\beta_1^{\ 2}-\nabla f(x_1)\nabla f(x_2)]^{\frac{1}{2}}$$

### 3.2.3 范例分析

考虑如下问题：求解 $x=[x1,x2]$，使得

$$\min_x f(x)=e^{x_1}(4x_1^{\ 2}+2x_2^{\ 2}+4x_1x_2+2x_2+1)$$

**【分析】**

为了求解此问题，可先利用 MATLAB 的文件编辑器编写一个.m 文件返回函数 $f(x)$的

值，然后调用优化工具箱无约束优化函数 fminunc 进行求解。

**【程序清单】**

例程 3-1 是求解的 MATLAB 代码。

例程 3-1

```
function y=objfun(x)
y=exp(x(1))*(3*x(1)^2+2*x(2)^2+3*x(1)*x(2)+2*x(2)+1);

x0=[-1,1];
%采用标准算法
options=optimset('largescale','off');
[x,fval,exitflag,output]=fminunc('objfun',x0,options)
```

**【结果输出】**

```
x =
        0.5000    -1.0000
fval =
        1.3031e-010
exitflag =
        1
output =
      iterations: 7
      funcCount: 30
      stepsize: 1
      firstorderopt: 8.1995e-003
      algorithm: 'medium-scale: Quasi-Newton line search'
```

## 3.3　约束优化算法及实现

约束优化问题根据约束函数的性质可分为：线性约束优化问题和非线性约束优化问题。其标准形式分别如下所示。

线性约束情形：

$$\min_{x} f(\boldsymbol{x})$$

$$\text{约束为：}\begin{cases} \boldsymbol{Ax} \leqslant \boldsymbol{b} \\ \boldsymbol{Aeq} * \boldsymbol{x} = \boldsymbol{beq} \end{cases}$$

非线性约束情形：

$$\min_{x} f(\boldsymbol{x})$$

$$\text{约束为：} g_i(\boldsymbol{x}) \leqslant 0,\ i = 1, \cdots, m$$

求解约束优化问题的思路主要分为两大类：一是直接对目标函数采用搜索法在可行方向求出最优解；另一个是对目标函数进行转换，化为更易求解的问题来求原优化问题的最优解，比如无约束优化问题或动态规划问题等。

### 3.3.1 可行方向法

可行方向法可以看做是无约束下降算法的自然推广，其典型策略是从可行点出发，沿着下降的可行方向进行搜索，求出使目标函数值下降的新的可行点。算法的主要步骤是选择搜索方向和确定沿此方向移动的步长。

### 3.3.2 惩罚函数法

惩罚函数法的基本思想就是，借助惩罚函数把约束问题转化为无约束问题，进而用无约束最优化方法来求解。由于约束的非线性，不能用消元法将问题化为无约束问题，因此在求解时必须同时照顾到既使目标函数值下降，又要满足约束条件这两个方面。实现这一点的一种途径是由目标函数和约束函数组成辅助函数，把原来的约束问题转化为极小化辅助函数的无约束问题。

考虑等式约束问题：

$$\min f(x) \qquad \text{（式 3.5）}$$

$$\text{约束为：} h_j(x)=0$$

可定义辅助函数：

$$F_1(x)=f(x)+\sigma\sum_{j=1}^{m}h_j^{\,2}(x)$$

其中参数σ是很大的正数，常称做惩罚因子。这样就把问题（式 3.5）转化为无约束问题：

$$\min F_1(x,\sigma) \qquad \text{（式 3.6）}$$

显然，（式 3.6）的最优解必定使 $h_j(x)$ 接近零，否则，（式 3.5）的第二项将是很大的正数。因此求解问题（式 3.6）能够得到问题（式 3.5）的近似解。

考虑不等式约束：

$$\min f(x) \qquad \text{（式 3.7）}$$

$$\text{约束为：} g_i(x)\geqslant 0$$

辅助函数的形式与等式约束情形不同，但构造辅助函数的基本思想是一致的，这就是在可行点辅助函数值等于原来的目标函数值；在不可行点，辅助函数值等于原来的目标函

数值加上一个很大的正数。由此，定义辅助函数：

$$F_2(x,\sigma)=f(x)+\sigma\sum_{i=1}^{m}[\max\{0,-g_i(x)\}]^2$$

当 $x$ 为可行点时，$\max\{0,-g_i(x)\}=0$；

当 $x$ 为不可行点时，$\max\{0,-g_i(x)\}=-g_i(x)$

这样，可将问题（式 3.7）转化为无约束问题

$$\min F_2(x,\sigma)$$

### 3.3.3 二次规划（QP）算法及实现

在约束条件极值问题中，常用的方法是先将原问题转化为较容易的子系统问题，然后再求解并用做一个迭代过程的基础。很多种约束优化的一个特点就是为约束条件增加一个惩罚函数，从而将约束问题转化为基本的无约束条件问题。按照这种方法，条件极值问题可以通过参数化无约束条件优化序列来求解，不过这样求解的效率不高。目前这种方法已经被集中于对 Kuhn-Tucker（KT）方程进行求解的方法所取代。KT 方程是条件极值问题的必要条件，如果要解决的问题是所谓的凸规划问题（也即 $f(x)$，$G_j(x)$ 都是凸函数），那么 KT 方程是条件极值问题的充分必要条件

Kuhn-Tucker（KT）方程可以表示为：

$$\nabla f(x^*)+\sum_{i=1}^{m}\lambda_i^*\nabla g_i(x^*)=0$$

$$\lambda_i^*\nabla g_i(x^*)=0,\quad i=1,\cdots,m_e$$

$$\lambda_i^*\geqslant 0,\quad i=m_{e+1},\cdots,m$$

上式第一个方程说明了优化目标函数和约束条件之间的梯度相互抵消，其中 Langrange 乘子$\lambda_i$（$i$=1,…,$m$）平衡了优化目标函数和约束条件之间梯度幅值大小的差异。

求解 KT 方程是很多非线性规划算法的基础，这些方法试图直接计算 Langrange 乘子 $\lambda_i$（$i$=1,…,$m$）。约束条件的拟牛顿法通过使用拟牛顿更新程序对 KT 方程累积二阶信息以保证超线性收敛。因为在每一个主要的迭代步骤中解决一个二次规划子问题，所以这些方法一般又被称为序列二次规划方法（Sequential Quadratic Programming，SQP），也称为迭代二次规划（Iterative Quadratic Programming，IQP）、回归二次规划方法（Recursive Quadratic Programming，RQP）等。下面将介绍解决约束优化问题的两种主要方法：序列二次规划方法（SQP）和二次规划（QP）子问题。

1．**算法描述**

序列二次规划（SQP）方法可以模拟解决无约束优化问题的牛顿方法来解决约束优化问题，在每一次迭代时，收敛可以由用拟牛顿（Quasi-Newton）方法得到的 Langrange 函

数构成的 Hessian 矩阵来保证，从而转化为一个二次规划（QP）子问题，它的解产生线性搜索过程的搜索方向。序列二次规划（SQP）方法的主要思想如下所述：

给定一个（式 3.8）所示的 GP 问题，它的求解的基本思想就是基于 Langrange 函数的二次近似求解子问题二次规划（QP）。

$$L(x,\lambda)=f(x)+\sum_{i=1}^{m}\lambda_i g_i(x), \qquad \text{（式 3.8）}$$

上式是假设约束条件为不等式约束后简化得到的，因此通过线性化非线性约束条件可以得到二次规划（QP）子问题：

$$\min_{d\in R^n}\frac{1}{2}d^T\boldsymbol{H}_k d+\nabla f(x_k)^T d$$
$$\nabla g_i(x)^T d+g_i(x)=0 \qquad i=1,\cdots,m_e$$
$$\nabla g_i(x)^T d+g_i(x)\leq 0 \qquad j=m_{e+1},\cdots,m$$

此子问题可以通过任何 QP 算法来求解，例如可形成如下形式的新迭代方程：

$$x_{k+1}=x_k+\alpha_k d_k$$

步长参数$\alpha_k$通过合适的线性搜索过程来确定，从而可以使得某一指标函数值得到足够的下降量。矩阵 $\boldsymbol{H}_k$ 是 Langrange 函数 Hessian 矩阵的正定近似，$\boldsymbol{H}_k$ 可以用任何拟牛顿法进行更新，但 BFGS 方法更常用一些。

使用 SQP 算法，求解一个非线性约束优化问题比一个无约束优化问题所迭代的次数要少，这是因为受到解可行区域的限制，这种方法更可能得到恰当的搜索方向和迭代步长。例如考虑如下带非线性不等式约束的 Rosenbrock 函数：

$$f(x)=100(x_2-x_1^2)^2+(1-x_1)^2$$
$$x_1^2+x_2^2-1.5\leq 0$$

求解此函数的极值，如果用 SQP 算法实现，需经过 96 次迭代后得到问题的解；如果当做一个无约束优化问题来求解，则需迭代 130 次才能得到问题的解。

**2．算法的.MATLAB 实现**

MATLAB 优化工具箱的 SQP 算法的实现由三部分组成。

（1）更新 Langrange 函数的 Hessian 矩阵

在每一次主迭代中，$\boldsymbol{H}$ 均作为 Langrange 函数的 Hessian 矩阵的正定拟牛顿近似，采用 BFGS 方法进行计算，其中$\lambda_i$（$i$=1,…,$m$）是 Langrange 乘子的估计。

Hessian 矩阵更新（采用 BFGS 方法）：

$$\boldsymbol{H}_{k+1}=\boldsymbol{H}_k+\frac{q_k q_k^T}{q_k^T s_k}-\frac{\boldsymbol{H}_k^T\boldsymbol{H}_k}{s_k^T\boldsymbol{H}_k s_k}$$

其中：

$$s_k = x_{k+1} - x_k$$

$$q_k = \nabla f(x_{k+1}) + \sum_{i=1}^{m} \lambda_i \nabla g_i(x_{k+1}) - (\nabla f(x_k) + \sum_{i=1}^{m} \lambda_i \nabla g_i(x_k))$$

$\boldsymbol{H}$ 初始值为一个正定矩阵，在更新过程中通过保证 $q_k{}^T s_k$ 的正定性来维持 Hessian 矩阵的正定性，当 $q_k{}^T s_k$ 非正定时，通过修改其中的元素使得 $q_k{}^T s_k > 0$。这样修改的目的是尽可能少地错改 $q_k$ 的值（$q_k$ 与 Hessian 矩阵的正定更新有直接联系），因此在修改的开始阶段，$q_k{}^T s_k$ 中的大多数负的元素不断被除以 2 直至 $q_k{}^T s_k$ 大于或等于 1e-5，如果在此过程中 $q_k{}^T s_k$ 仍然为负，则通过加上一个向量 $\boldsymbol{v}$ 与常数标量 $w$ 的乘积来改变 $q_k$ 的值。

$$q_k = q_k + vw$$

其中：

$$v_i = \begin{cases} \nabla g_i(x_{k+1}) * g_i(x_{k+1}) - \nabla g_i(x_k) * g_i(x_k), (q_k)_i * (s_k)_i < 0 and (q_k)_i * w < 0 \\ 0, \quad 其他 \end{cases}$$

$w$ 对称增加直至 $q_k{}^T s_k$ 为正。

在 MATLAB 优化工具箱中，fmincon、fminimax、fgoalattain 和 fseminf 都使用了 SQP 算法。

（2）求解二次规划子问题

在 SQP 算法的每一次主迭代中都要求解一次如（式 3.9）所示的 QP 问题

$$\min_{x \in R} \frac{1}{2} \boldsymbol{x}^T \boldsymbol{H} \boldsymbol{x} + c^T \boldsymbol{x} \text{，} \qquad （式 3.9）$$

满足：

$$\begin{cases} \boldsymbol{A}_i * \boldsymbol{x} = \boldsymbol{b}, & i = 1, \cdots, m_{e+1} \\ \boldsymbol{A}_i * \boldsymbol{x} \leqslant \boldsymbol{b}, & i = m_e, \cdots, m \end{cases}$$

求解过程包含两个阶段：第一阶段计算解的一个可行点；第二阶段产生可行点的一个迭代序列，这个序列收敛到问题的解。在这种方法中，一直保持着一个活动集合 $\bar{\boldsymbol{A}}_k$，$\boldsymbol{A}_k$ 在每一次迭代时被更新，从而构成搜索方向 $\hat{d}_k$ 的基础，等式约束的信息也包含在 $\bar{\boldsymbol{A}}_k$ 中，在搜索方向 $\hat{d}_k$ 上，目标函数的值在约束边界内取得最小值；而 $\hat{d}_k$ 的可行子空间又以

$\boldsymbol{Z}_k$ 为基础，$\boldsymbol{Z}_k$ 与 $\bar{\boldsymbol{A}}_k$ 的估计值是正交的，即 $\bar{\boldsymbol{A}}_k * \boldsymbol{Z}_k = 0$，因此由 $\boldsymbol{Z}_k$ 任意列的线性和所构成的搜索方向被保证在约束边界内。

矩阵 $\boldsymbol{Z}_k$ 由矩阵 $\bar{\boldsymbol{A}}_k^T$ QR 分解后的第 $l$+1 列到第 $m$ 列组成，$l$ 是约束的数目（$l<m$），也就是说由下式得到：

$$\boldsymbol{Z}_k = \boldsymbol{Q}(l+1:m)$$

其中：

$$\boldsymbol{Q}^T \bar{\boldsymbol{A}}_k^t = \begin{bmatrix} R \\ 0 \end{bmatrix}$$

这样，我们以 $p$ 代替 $\hat{d}_k$，则待求的二次问题可以转化为 $p$ 的函数：

$$q(p) = \frac{1}{2} p^T \boldsymbol{Z}_k^t \boldsymbol{H} \boldsymbol{Z}_k p + c^T \boldsymbol{Z}_k p$$

关于 $p$ 差分形式为：

$$\nabla q(p) = \boldsymbol{Z}_k^T \boldsymbol{H} \boldsymbol{Z}_k p + \boldsymbol{Z}_k^T c$$

$\nabla q(p)$ 称为二次函数的规划梯度，$\boldsymbol{Z}_k^T \boldsymbol{H} \boldsymbol{Z}_k$ 称为规划 Hessian 矩阵，当 $\nabla q(p) = 0$ 时，$q(p)$ 在由 $\boldsymbol{Z}_k$ 定义的子空间上取得最小值，它就是下列线性方程的解：

$$\boldsymbol{Z}_k^T \boldsymbol{H} \boldsymbol{Z}_k p = -\boldsymbol{Z}_k^T c$$

从而得到迭代形式：

$$x_{k+1} = x_k + \alpha \hat{d}_k \text{，其中：} \hat{d}_k = Z_k^T * p$$

（3）线性搜索和计算指标函数

如上节中所述，求解 QP 子问题得到一个向量 $d_k$，由它可得到新的迭代：

$$x_{k+1} = x_k + \alpha_k * d_k$$

$\alpha_k$ 的每次取值必须保证指标函数有足够的下降量，这里的指标函数如（式 3.10）所示：

$$\psi(x) = f(x) + \sum_{i=1}^{m_e} r_i * g_i(x) + \sum_{i=m_e+1}^{m} r_i * \max\{0, g_i(x)\} \text{，} \qquad \text{（式 3.10）}$$

其中：

$$r_i = (r_{k+1})_i = \max_i \{\lambda_i * \frac{1}{2}((r_k)_i + \lambda_i)\}, \quad i = 1, \cdots, m$$



在实际应用中，有许多与（式 3.10）形式很相似的问题，这时 MATLAB 本身没有现成的函数可以调用，但是可以根据优化算法编写.m 文件来实现。

### 3.3.4　范例分析

这部分是有约束优化问题的几个例子，并给出了详细的 MATLAB 源代码。

**1．非线性不等式约束**

已知函数 $f(x)=e^{x_1}*(4x_1^2+2x_2^2+4x_1x_2+2x_2+1)$，且满足非线性约束：

$$\begin{cases} x_1x_2-x_1-x_2\leqslant -1.5 \\ x_1x_2\geqslant -10 \end{cases}，求 \min f(x)$$

**【分析】**

这样的非线性约束优化问题可以用 MATLAB 优化工具箱中的 fmincon 函数来求解，因为约束为非线性约束，所以不可能将约束条件信息直接包含在函数的输入参数中，必须编写函数返回在每一个点 $x$ 处的约束值，然后再调用优化函数 fmincon；另一方面，fmincon 函数要求的约束条件一般为 $c(x)\leqslant 0$，所以将约束改为：

$$\begin{cases} x_1x_2-x_1-x_2+1.5\leqslant 0 \\ -x_1x_2-10\leqslant 0 \end{cases}$$

**【程序清单】**

例程 3-2 是求解的 MATLAB 代码。

**例程 3-2**

```
%编写目标函数
function y=objfun(x)
y=exp(x(1))*(3*x(1)^2+2*x(2)^2+3*x(1)*x(2)+2*x(2)+1);

%编写返回约束值的函数
function [c,ceq]=confun(x)
%非线性不等式约束
c=[1.5+x(1)*x(2)-x(1)-x(2);-x(1)*x(2)-10];
%线性等式约束
ceq=[];

x0=[-1,1];
%采用标准算法
options=optimset('largescale','off');
[x,fval]=fmincon('objfun',x0,[],[],[],[],[],[],'confun',options)
```

【结果输出】

```
x =
        -9.5373      1.0373
fval =
        0.0236
```

此时的约束值为：

```
c =
        1.0e-013 *
        0.1110
        -0.1776
ceq =
        []
```

输出函数值计算的总次数：

```
options（10）
```

得到的结果为：

```
ans=
    29
```

2. **边界约束问题**

已知函数 $f(x)=e^{x_1}*(4x_1^{\ 2}+2x_2^{\ 2}+4x_1x_2+2x_2+1)$，求 $\min\limits_{x} f(x)$

且满足非线性约束：

$$\begin{cases} x_1x_2-x_1-x_2\leqslant -1.5 \\ x_1x_2\geqslant -10 \end{cases}$$

边界约束：

$$\begin{cases} x_1\geqslant 0 \\ x_2\geqslant 0 \end{cases}$$

【分析】

此问题在非线性约束的基础上增加了变量 $x$ 的边界条件，在 fmincon 函数输入参数中加上 *lb* 和 *ub* 参数即可。

【程序清单】

例程 3-3 是求解的 MATLAB 代码。

例程 3-3

```
%编写目标函数
function y=objfun(x)
y=exp(x(1))*(3*x(1)^2+2*x(2)^2+3*x(1)*x(2)+2*x(2)+1);

%编写返回约束值的函数
function [c,ceq]=confun(x)
```

```
%非线性不等式约束
c=[1.5+x(1)*x(2)-x(1)-x(2);-x(1)*x(2)-10];
%线性等式约束
ceq=[];

%初始点
x0=[-1,1];
%设置下界
lb=[0,0];
%无上界
ub=[];
%采用标准算法
options=optimset ('largescale','off');
[x,fval]=fmincon('objfun',x0,[],[],[],[],lb,ub,'confun',options)
[c,ceq]=confun(x)
```

**【结果输出】**

```
x =
        0      1.5000
fval =
        8.5000
c =
        0
        -10
ceq =
        []
```

### 3．利用梯度求解约束优化问题

已知函数 $f(x)=e^{x_1}*(4x_1^{\ 2}+2x_2^{\ 2}+4x_1x_2+2x_2+1)$，且满足非线性约束：

$$\begin{cases} x_1x_2-x_1-x_2\leqslant -1.5 \\ x_1x_2\geqslant -10 \end{cases}，求 \min f(x)$$

**【分析】**

一般来说，标准算法求解函数的最小值常使用由有限差分逼近得到的离散数字梯度，在这个过程中，为了计算目标函数和约束的偏微分，所有变量被系统地加以调动。当使用梯度求解上述问题时，效率更高并且结果更精确。

题中目标函数的偏微分为：

$$\frac{\partial f}{\partial x}=\begin{bmatrix} e^{x_1}*(4x_1^{\ 2}+2x_2^{\ 2}+4x_1x_2+2x_2+1)+e^{x_1}*(8x_1+4x_2) \\ e^{x_1}*(4x_2+4x_1+2) \end{bmatrix}$$

**【程序清单】**

例程 3-4 是求解的 MATLAB 代码。

例程 3-4

```
%编写目标函数和梯度的.m 文件
function [f,g]=objfun(x)
f=exp(x(1))*(3*x(1)^2+2*x(2)^2+3*x(1)*x(2)+2*x(2)+1);
t=exp(x(1))*(3*x(1)^2+2*x(2)^2+3*x(1)*x(2)+2*x(2)+1);
%g 中包含着目标函数的偏微分信息
g=[t+exp(x(1))*(8*x(1)+3*x(2)),exp(x(1))*(3*x(1)+3*x(2)+2)];
%编写不等式约束及其梯度的.m 文件

function [c, ceq,dc,dceq]=confun(x)
%不等式约束
c=[1.5+x(1)*x(2)-x(1)-x(2);-x(1)*x(2)-10];
%约束的梯度
dc=[x(2)-1,-x(2);x(1)-1,-x(1)];
%没有非线性等式约束
ceq=[];
dceq=[];
%dc 的列包含着不同约束各自的偏微分，也就是说，dc 的第 i 列是第 i 个约束对 x 的
%偏微分，在此处 dc 为：
```

% $$\begin{bmatrix} \dfrac{\partial c_1}{\partial x_1} & \dfrac{\partial c_2}{\partial x_1} \\ \dfrac{\partial c_1}{\partial x_2} & \dfrac{\partial c_2}{\partial x_2} \end{bmatrix} = \begin{bmatrix} x_2-1 & -x_2 \\ x_1-1 & -x_1 \end{bmatrix}$$

```
x0=[-1,1];
lb=[];
ub=[];
%采用标准算法
options=optimset ('largescale','off');
%采用梯度
options=optimset (options,'GradObj','on','GradConstr','on');
[x,fval]=fmincon('objfun',x0,[],[],[],[],lb,ub,'confun',options)
[c,ceq]=confun(x)
```

**【结果输出】**

```
x =
        -9.5373      1.0373
fval =
        0.0236
c =
        1.0e-013 *
        0.1110
        -0.1776
ceq =
        []
```



函数计算总次数为：11，显然比不使用梯度时的计算量 29 次要小得多。

4．等式约束优化

已知函数 $f(x)=e^{x_1}*(4x_1^2+2x_2^2+4x_1x_2+2x_2+1)$，且满足非线性约束：

$$\begin{cases} x_1x_2-x_1-x_2\leqslant -1.5 \\ x_1x_2\geqslant -10 \\ x_1^2+x_2=1 \end{cases}，求 \min_x f(x)$$

【分析】

此问题在前述问题的基础上增加了一个等式约束，等式约束的信息可以含在MATLAB优化工具箱函数的输入参数：矩阵 ***Aeq*** 和向量 ***b*** 中，为了符合优化函数的调用形式，将约束转化为：

$$\begin{cases} 1.5+x_1x_2-x_1-x_2\leqslant 0 \\ -x_1x_2-10\leqslant 0 \\ x_1^2+x_2-1=0 \end{cases}$$

【程序清单】

例程 3-5 是求解的 MATLAB 代码。

例程 3-5

```
%编写目标函数
function f=objfun(x)
f=exp(x(1))*(3*x(1)^2+2*x(2)^2+3*x(1)*x(2)+2*x(2)+1);

%编写非线性约束的.m 文件
function [c,ceq]=confun(x)
%不等式约束
c=[1.5+x(1)*x(2)-x(1)-x(2);-x(1)*x(2)-10];
%等式约束
ceq=x(1)^2+x(2)-1;

%初始点
x0=[-1,1];
%无边界约束
lb=[];
ub=[];
%采用标准算法
options=optimset('largescale','off');
[x,fval]=fmincon('objfun',x0,[],[],[],[],lb,ub,'confun',options)
[c,ceq]=confun(x)
```

【结果输出】

```
x =
        0.1578      1.3110
fval =
        10.2971
c =
        0.1539
        -10.2227
ceq =
        0.3359
```

5. 非线性滤波器的优化设计

现有一个三阶低通切比雪夫滤波器，截止频率 $Wn$ = 0.2kHz，通带内波纹 $Rp$ = 1.5db，其频率响应如图 3-2 所示。现要求设计一个与上述指标相似的滤波器，且它的系数必须取整数值。

【分析】

对于滤波器的设计问题，.MATLAB 的信号处理工具箱中有很多函数可以调用，本问题中调用 cheby1 函数就可以设计出满足要求的低通切比雪夫滤波器，但是它的系数为非整数量。因此，本问题的实质是一个以滤波器系数为优化变量，使得设计出的滤波器与给定指标最接近且要求优化变量取离散值的优化设计问题。这类问题可以通过求解一个等价的连续问题来解决，在求解过程中首先对第一个变量的值进行上下取整运算得到最近且最优的离散值，从而解决一个变量的离散化问题。如此类推，当所有的变量都取到最优的离散值后，这类问题也就得到解决了。

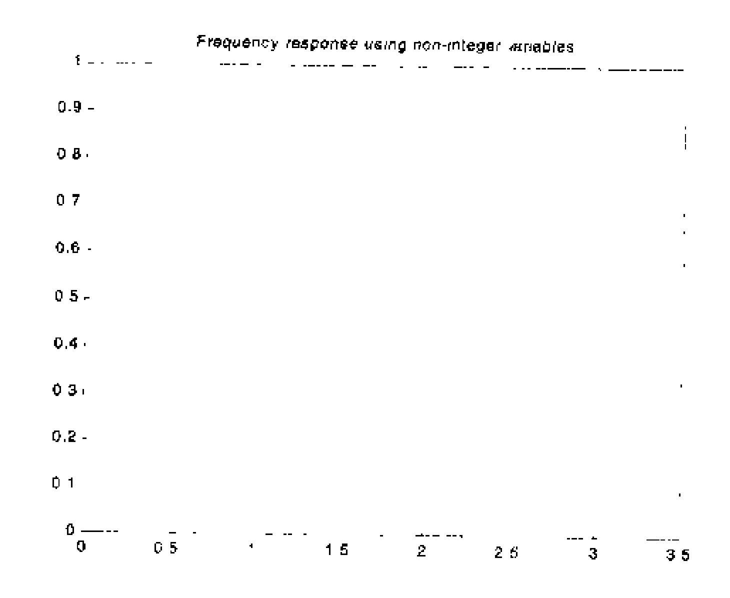



图 3-2 优化前的滤波器的频谱图

【程序清单】

例程 3-6 是求解的 MATLAB 代码。

例程 3-6

```
nbits = 8;        % 滤波器阶数
maxbin = 2^nbits-1;
n=3;                % 滤波器的阶数为 n-1
Wn = 0.2;         % 滤波器的截止频率
Rp = 1.5;         % 通带内波纹的分贝值
w = 128;          % 频率点数目
% 连续切比雪夫滤波器设计
[b1,a1]=cheby1(n-1,Rp, Wn);
[h,w]=freqz(b1,a1,w);    % 频率响应
h=abs(h);         % 幅频响应
plot(w, h);
title ('Frequency response using non-integer variables')
x = [b1,a1];          % 设计变量

% 设置 x 的最大最小边界值
if (any(x < 0))
     maxbin = floor (maxbin/2);
     vlb = -maxbin * ones(1, 2*n)-1;
     vub = maxbin * ones(1, 2*n);
else
     vlb = zeros(1,2*n);
     vub = maxbin * ones(1, 2*n);
end

%归一化滤波器系数
[m, mix] = max(abs(x));
factor =   maxbin/m;
x =   factor * x;      %
xorig = x;
xmask = 1:2*n;
xmask([mix]) = [];
nx = 2*n;
% 设置算法终止准则，加速解的收敛
options = optimset('Display','iter','TolX',0.1,...
'TolFun',1e-3,'TolCon',1e-6);
if length(w) == 1
    options = optimset(options,'MinAbsMax',w);
else
    options = optimset(options,'MinAbsMax',length(w));
end
%离散化第一个变量
[x, xmask] = elimone(x, xmask, h, w, n, maxbin)

niters = length(xmask);
for m = 1:niters
```

```
%计算最优的离散整数值
x(xmask) = fminimax('filtobj',x(xmask),[],[],[],[],vlb(xmask),...
vub(xmask),'filtcon',options, x, xmask, n, h, maxbin);
[x, xmask] = elimone(x, xmask, h, w, n, maxbin);
end

xold = x;
xmask = 1:2*n;
xmask([n+1, mix]) = [];
x = x + 0.5;
for i = xmask
    [x, xmask] = elimone(x, xmask, h, w, n, maxbin);
end
xmask = 1:2*n;
xmask([n+1, mix]) = [];
x= x - 0.5;
for i = xmask
    [x, xmask] = elimone(x, xmask, h, w, n, maxbin);
end
if any(abs(x) > maxbin)
x = xold;
end

% 优化滤波器的频率响应
subplot(211)
bo = x(1:n);
ao = x(n+1:2*n);
h2 = abs(freqz(bo,ao,128));
plot(w,h,w,h2,'o')
title('Optimized filter versus original')
```

**【结果输出】**

滤波器系数如表 3-1 所示。

表 3-1　滤波器系数

| | b0 | b1 | b2 | b3 | a0 | a1 | a2 | a3 |
|---|---|---|---|---|---|---|---|---|
| 优化前系数 | 0.0095 | 0.0286 | 0.0286 | 0.0095 | 1.0000 | -2.2216 | 1.8895 | -0.5917 |
| 优化后系数 | 0 | 2 | 2 | 0 | 57 | -127 | 108 | -33 |

优化后的滤波器的频谱图如图 3-3 所示（取 128 个频率点）。图 3-3 与图 3-2 相比很相似，因而满足设计要求。

**6．求解下列函数的最小值**

$$\min_x f(x) = \min_x(x_1^2 + 2 * x_2^2 + x_3^2 - 2 * x_1 * x_2 + x_3)$$



约束为：$\begin{cases} x_1 + x_2 + x_3 = 4 \\ 2 * x_1 - x_2 + x_3 = 2 \end{cases}$

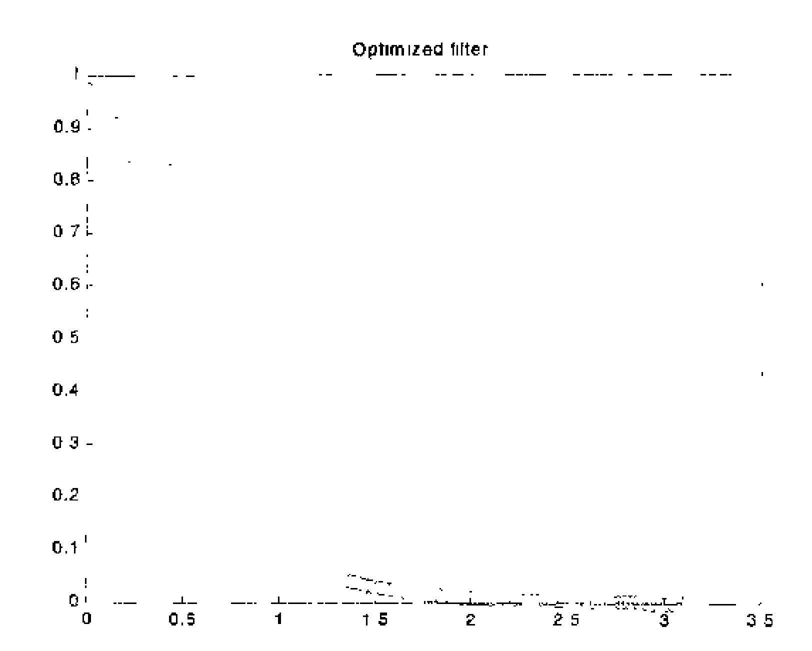



图 3-3　优化后的滤波器的频谱图

【分析】

根据目标函数的形式，它可以写成二次型函数的形式，即：

$$f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T \boldsymbol{H} \boldsymbol{x} + \boldsymbol{c}^T \boldsymbol{x}$$

其中：

$$\boldsymbol{H} = \begin{bmatrix} 2 & -2 & 0 \\ -2 & 4 & 0 \\ 0 & 0 & 2 \end{bmatrix}, \quad \boldsymbol{c} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad \boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

约束为：

$$\boldsymbol{Aeq} * \boldsymbol{x} = \boldsymbol{beq}$$

其中：

$$\boldsymbol{Aeq} = \begin{bmatrix} 1 & 1 & 1 \\ 2 & -1 & 1 \end{bmatrix}, \quad \boldsymbol{beq} = \begin{bmatrix} 4 \\ 2 \end{bmatrix}$$

所以可以采用二次规划函数 quadprog 来求解。

【程序清单】

例程 3-7 是求解的 MATLAB 代码。

例程 3-7

```
%系数
H=[2 -2 0;-2 3 0;0 0 2];
c=[0 0 1]';
Aeq=[1 1 1;2 -1 1];
beq=[3,2]';
%无等式约束
A=[];
b=[];
[x,fval]=quadprog(H,c,A,b,Aeq,beq)
```

【结果输出】

```
x =
        1.9091
        1.9535
        0.1363
fval =
        3.9773
```

7. 求解下列形式的问题的最小值

$$\min_x f(\boldsymbol{x}) = \min_x \left(\frac{1}{2}\boldsymbol{x}^T \boldsymbol{G}\boldsymbol{x} + \boldsymbol{c}^T\boldsymbol{x} + gamma\right)$$

约束为：$\boldsymbol{lb} \leqslant \boldsymbol{x} \leqslant \boldsymbol{ub}$

其中：$\boldsymbol{G}$ 是一个 $n\times n$ 的对称矩阵。

【分析】

显然，目标函数的形式与二次型函数很相似，但不能转化为二次规划问题的标准形式，所以不可能直接调用优化函数，这里采用求解二次规划问题的算法来求解这一类问题。

【程序清单】

例程 3-8 是求解的 MATLAB 代码：minq.m。

例程 3-8

```
function [x,fct,ier,nsub]=minq(gam,c,G,xu,xo,prt,xx);
%%% 初始化 %%%
if prt>0, printlevel=prt, end;
convex=0;
n=size(G,1);
% 最大迭代次数
maxit=3*n;
% 设置初始点
if nargin<7,
```

```
        xx=zeros(n,1);
    end;
    % 初始点位于界限内
    xx=max(xu,min(xx,xo));
    % 规范化低阶子问题
    hpeps=100*eps;
    G=G+spdiags(hpeps*diag(G),0,n,n);

    K=logical(zeros(n,1));
    if issparse(G), L=speye(n); else L=eye(n); end;
    dd=ones(n,1);
    free=logical(zeros(n,1));
    nfree=0;
    nfree_old=-1;

    fct=inf;          % 函数最优值
    nsub=0;           % 子空间的数目
    unfix=1;          % 允许变量自由变化
    nitref=0;
    improvement=1;

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % 主循环：进行坐标和子空间搜索
    while 1,
       if prt>1, disp('enter main loop'); end;
       if norm(xx,inf)==inf, error('infinite xx in minq.m'); end;
       g=G*xx+c;
       fctnew=gam+0.5*xx'*(c+g);
       if ~improvement,
        % good termination
        if prt,
          disp('terminate: no improvement in coordinate search');
        end;
        ier=0; break;
       elseif nitref>nitrefmax,
        % good termination
        if prt, disp('terminate: nitref>nitrefmax'); end;
        ier=0; break;
        elseif nitref>0 & nfree_old==nfree & fctnew >= fct,
        % good termination
        if prt,
          disp('terminate: nitref>0 & nfree_old==nfree & fctnew>=fct');
        end;
        ier=0; break;
        elseif nitref==0,
        x=xx;
```

```
fct=min(fct,fctnew);
if prt>1, fct, end;
if prt>2, X=x', fct, end;
else % more accurate g and hence f if nitref>0
x=xx;
fct=fctnew;
if prt>1, fct, end;
if prt>2, X=x', fct. end;
end;
if nitref==0 & nsub==maxit,
 if prt,
   disp('!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!');
  disp('!!!!!                minq              !!!!!');
  disp('!!!!! incomplete minimization !!!!!');
  disp('!!!!!     too many iterations      !!!!!');
  disp('!!!!!        increase maxit          !!!!!');
  disp('!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!');
else
  disp('iteration limit exceeded');
end;
ier=99;
break;
end;

% 坐标搜索
count=0;
k=0;
while 1,
  while count<=n,
    count=count+1;
    if k==n, k=0; end;
    k=k+1;
    if free(k) | unfix, break; end;
  end;
  if count>n,
    break;
  end;
  q=G(:,k);
  alpu=xu(k)-x(k); alpo=xo(k)-x(k); % bounds on step

  % 寻找步长大小
  [alp,lba,uba,ier]=getalp(alpu,alpo,g(k),q(k));
  if ier,
    x=zeros(n,1);
    if lba, x(k)=-1; else x(k)=1; end;
    if prt,
```

```
        gTp=g(k),pTGp=q(k),quot=pTGp/(norm(p,1)^2*norm(G(:),inf))
        disp('minq: function unbounded below in coordinate direction');
        disp('        unbounded direction returned');
        disp('        possibly caused by roundoff');
      end;
      if prt>1,
        disp('f(alp*x)=gam+gam1*alp+gam2*alp^2/2, where');
        gam1=c'*x
        gam2=x'*(G*x)
        ddd=diag(G);
        min_diag_G=min(ddd)
        max_diag_G=max(ddd)
      end;
      return;
    end;
    xnew=x(k)+alp;
    if prt & nitref>0,
      xnew,alp
    end;

    if lba | xnew<=xu(k),
      % 下界约束
      if prt>2, disp([num2str(k), ' at lower bound']); end;
      if alpu~=0,
        x(k)=xu(k);
        g=g+alpu*q;
        count=0;
      end;
      free(k)=0;
    elseif uba | xnew>=xo(k),
      % 上界约束
      if prt>2, disp([num2str(k), ' at upper bound']); end;
      if alpo~=0,
        x(k)=xo(k);
        g=g+alpo*q;
        count=0;
      end;
      free(k)=0;
    else
      % no bound active
      if prt>2, disp([num2str(k), ' free']); end;
      if alp~=0,
        if prt>1 & ~free(k),
          unfixstep=[x(k),alp],
        end;
        x(k)=xnew;
```

```
        g=g+alp*q;
        free(k)=1;
      end;
    end;

  end;
  % end of coordinate search

  nfree=sum(free);
  if (unfix & nfree_old==nfree),
    g=G*x+c;
    nitref=nitref+1;
    if prt>0,
      disp('optimum found; iterative refinement tried');
    end;
  else
    nitref=0;
  end;
  nfree_old=nfree;
  gain_cs=fct-gam-0.5*x'*(c+g);
  improvement=(gain_cs>0 | ~unfix);

  if prt,
    nfree=pr01('csrch ',free);
  end;
  if prt, gain_cs, end;
  if prt>2, X=x', end;

  % 子空间搜索
  xx=x;
  if ~improvement | nitref>nitrefmax,
    % optimal point found - nothing done
  elseif nitref>nitrefmax,
  elseif nfree==0,
    if prt>0,
      disp('no free variables - no subspace step taken')
    end;
    unfix=1;
  else
    minqsub;
    if ier, return; end;
  end;

  if prt>0,
    nfree=pr01('ssrch ',free);
    disp(' ');
```

```
        if unfix & sum(nfree)<n,
          disp('bounds may be freed in next csearch');
        end;
      end;
    end;
    % 结束主循环
    if prt>0,
      fct
      disp;
    end;
```

例程 3-9 是 getalp.m 的 MATLAB 代码。

例程 3-9

```
function [alp,lba,uba,ier]=getalp(alpu,alpo,gTp,pTGp);

lba=0;
uba=0;
% determine unboundedness
ier=0;
if alpu==-inf & ( pTGp<0 | (pTGp==0 & gTp>0) ),
  ier=1; lba=1;
end;
if alpo==inf & (pTGp<0 | (pTGp==0 & gTp<0) ),
  ier=1; uba=1;
end;
if ier, alp=NaN; return; end;

% determine activity
if pTGp==0 & gTp==0,
  alp=0;
elseif pTGp<=0,
  % concave case minimal at a bound
  if alpu==-inf,        lba=0;
  elseif alpo== inf, lba=1;
  else                    lba = (2*gTp+(alpu+alpo)*pTGp>0);
  end;
  uba = ~lba;
else
  alp=-gTp/pTGp;              % unconstrained optimal step
  lba = (alp <= alpu);     % lower bound active
  uba = (alp >= alpo);     % upper bound active
end;

if lba, alp=alpu; end;
if uba, alp=alpo; end;
```

例程 3-10 是 pr01.m 的 MATLAB 代码。

例程 3-10

```
function summe=pr01(name,x)
%检查行和列向量
[m,n]=size(x);row=(m==1);col=(n==1);n=max(m,n);
if ~(row|col), error('x must be a vector'); end;

text=[name,': '];summe=0;
for k=1:n,
   if x(k),
      text=[text,'1'];
      summe=summe+1;
   else
      text=[text,'0'];
   end;
end;
dip([text,'      ',num2str(summe),' nonzeros']);
```

8．求解下列形式的最优化问题

$$\min_x f(\boldsymbol{x}) = \min_x(\frac{1}{2}\boldsymbol{x}^T\boldsymbol{G}\boldsymbol{x} + c^T\boldsymbol{x})$$

约束为：$\boldsymbol{A} * \boldsymbol{x} \geqslant \boldsymbol{b}$

其中：$\boldsymbol{G}$ 是一个 $n \times n$ 的正定对称矩阵。

**【分析】**

这一类问题可以直接调用 MATLAB 优化工具箱中的 quadprog 函数；这里采用求解二次规划问题的算法，调用上述函数 minq 来求解这一类问题。

**【程序清单】**

例程 3-11 是求解的 MATLAB 代码：minqdef.m。

例程 3-11

```
function [x,y,ier]=minqdef(c,G,A,b,eq,prt,xx);
R=chol(G);
[m,n]=size(A);
A0=A/R;
GG=A0*A0';
c0=R'\c;
cc=-b-A0*c0;
yo=inf+zeros(m,1);
yu=zeros(m,1);yu(eq)=-yo(eq);

[y,fct,ier]=minq(0,cc,GG,yu,yo,prt);
x=R\(A0'*y-c0);
```

```
if ier==99, return; end;

% check for accuracy
res=A*x-b;ressmall=nnz(A)*eps*(abs(A)*abs(x)+abs(b));
res(~eq)=min(res(~eq),0);
if prt,
      disp('residual (first row) small if comparable to second row')
      disp([res,ressmall]')
end;
if min(abs(res)<=ressmall),
      % accuracy satisfactory
      ier=0;
      return;
end;

% one step of iterative refinement
if prt,
      disp('one step of iterative refinement')
end;
[dy,fct,ier]=minq(0,-res,GG,yu-y,yo-y,prt);
x=x+R\(A0'*dy);
y=y+dy;

% check for accuracy
res=A*x-b;ressmall=nnz(A)*eps*(abs(A)*abs(x)+abs(b));
res(~eq)=min(res(~eq),0);
if min(abs(res)<=sqrt(nnz(A))*ressmall),
      % accuracy satisfactory
      ier=0;
else
      % feasible set probably empty
      ier=1;
end;
```

**9．利用范例 7 和 8 的函数求解下列优化问题**

$$\min_x f(\boldsymbol{x})=\min_x(\frac{1}{2}\boldsymbol{x}^T\boldsymbol{G}\boldsymbol{x}+c^T\boldsymbol{x})$$

约束为：$\boldsymbol{A}*\boldsymbol{x}\geq\boldsymbol{b}$

**【程序清单】**

例程 3-12 是求解的 MATLAB 代码。

例程 3-12

```
n=7;
m=10;
p=0.8;
```

```
    % 生成满足 KKT 条件的随机数据
    A=rand(m,n);
    c=rand(n,1);
d=rand(n,1);
G=diag(d);
ydes=rand(m,1)-p;
act=( ydes>p );
ydes(act)=0*ydes(act);
eq=( ydes<0 );
xdes=(A'*ydes-c)./d;
res=rand(m,1);
res(~act)=0*res(~act);
b=A*xdes-res;
prt=0;

disp('test of minqdef.m');
[x,y,ier]=minqdef(c,G,A,b,eq,prt);
```

【结果输出】

```
test of minqdef.m
A =
    0.8030    0.9613    0.3333    0.2731    0.3290    0.7015    0.8699
    0.0839    0.0721    0.5625    0.6262    0.3782    0.0922    0.7693
    0.9355    0.5533    0.6166    0.5369    0.5972    0.3239    0.3332
    0.9159    0.2920    0.1133    0.0595    0.1613    0.3756    0.6206
    0.6020    0.8580    0.8983    0.0890    0.8295    0.1662    0.9517
    0.2536    0.3358    0.7536    0.2713    0.9561    0.8332    0.6300
    0.8735    0.6802    0.7911    0.3091    0.5955    0.8386    0.2373
    0.5133    0.0533    0.8150    0.3730    0.0287    0.3516    0.3527
    0.7327    0.3567    0.6700    0.9090    0.8121    0.9566    0.1879
    0.3222    0.3983    0.2009    0.5962    0.6101    0.1372    0.3906
c =
    0.3093
    0.3635
    0.6109
    0.0712
    0.3133
    0.6083
    0.1750
d =
    0.6210
    0.2360
    0.5873
    0.5061
    0.3638
    0.5313
    0.9323
```

```
G =
    0.6210         0         0         0         0         0         0
         0    0.2360         0         0         0         0         0
         0         0    0.5873         0         0         0         0
         0         0         0    0.5061         0         0         0
         0         0         0         0    0.3638         0         0
         0         0         0         0         0    0.5313         0
         0         0         0         0         0         0    0.9323
b =
  -25.0967
  -11.8157
  -22.3523
  -12.8302
  -25.3320
  -21.1701
  -25.2735
  -12.2063
  -23.1370
  -16.3726
y =
   -0.3175
   -0.3337
   -0.3732
   -0.3051
   -0.3305
   -0.3352
   -0.3363
   -0.0732
   -0.3673
   -0.7651
 x =
   -3.8231
  -10.6562
   -3.9665
   -3.7671
   -6.0812
   -3.8078
   -2.7803
ier =
     0
```

## 3.4　最小二乘优化算法及实现

在一个最小二乘优化问题中，目标函数 $f(x)$是一个平方和的形式，并求其最小值，如下式所示：

$$\min_{x\in R^n} f(x)=\frac{1}{2}\|F(x)\|_2^2=\frac{1}{2}\sum_{i=1} F_i(x)^2$$

这样的问题在实际应用中经常遇到。根据目标函数的性质，最小二乘优化问题通常可以分为两大类：线性最小二乘优化和非线性最小二乘优化问题，其形式如下所示。

（1）线性最小二乘优化问题

$$\min_x f(\boldsymbol{x})=\min_x \frac{1}{2}\|\boldsymbol{C}*\boldsymbol{x}-\boldsymbol{d}\|_2^2 \qquad \text{（式 3.11）}$$

约束可能为：

$$\boldsymbol{A}*\boldsymbol{x}\leqslant \boldsymbol{b}$$
$$\boldsymbol{Aeq}*\boldsymbol{x}=\boldsymbol{beq}$$
$$\boldsymbol{lb}\leqslant \boldsymbol{x}\leqslant \boldsymbol{ub}$$

其中 $\boldsymbol{C}$，$\boldsymbol{A}$，$\boldsymbol{Aeq}$ 为矩阵；$\boldsymbol{b}$，$\boldsymbol{beq}$ 为向量。

.MATLAB 优化工具箱中的 lsqlin 函数可用于求解线性最小二乘优化问题。

（2）非线性最小二乘优化问题

这类问题常用于将给定数据按照给定的函数族进行拟合，即按非线性参数估计问题。最小二乘优化还常用于控制问题：使得输出 $y(x,t)$跟踪某个期望的连续轨迹$\phi(t)$。这种问题可以表示为（式 3.12）

$$\min_{x\in R^n}\int_{t_1}^{t_2}(y(x,t)-\phi(t))\mathrm{d}t\,, \qquad \text{（式 3.12）}$$

其中：$y(x,t)$和$\phi(t)$均是向量函数。

利用合适的方法，（式 3.12）可以被离散化成（式 3.13）所示的最小二乘问题：

$$\min_{x\in R^n} \boldsymbol{f}(x)=\sum_{i=1}^{m}(\bar{\boldsymbol{y}}(x,t_i)-\bar{\boldsymbol{\phi}}(t_i)), \qquad \text{（式 3.13）}$$

按照最小二乘优化的一般形式，此处的 $\boldsymbol{F}(x)$为：

$$\boldsymbol{F}(x)=\begin{bmatrix}\bar{\boldsymbol{y}}(x,t_1)-\bar{\boldsymbol{\phi}}(t_1)\\ \bar{\boldsymbol{y}}(x,t_2)-\bar{\boldsymbol{\phi}}(t_2)\\ \vdots\\ \bar{\boldsymbol{y}}(x,t_m)-\bar{\boldsymbol{\phi}}(t_m)\end{bmatrix}$$

这一类问题虽然可以用前面的无约束优化方法来求解，但是这类问题的特殊形式又使得可以寻求另外更有效的优化方法，提高求解过程的迭代效率，现分析如下。

最小二乘问题的梯度和 Hessian 矩阵有特殊的结构，记 $\boldsymbol{J}(x)$为 $\boldsymbol{F}(x)$的 Jacobian 矩阵、$\boldsymbol{g}(x)$为$\boldsymbol{f}(x)$的梯度向量、$\boldsymbol{H}(x)$为$\boldsymbol{f}(x)$的 Hessian 矩阵，则可以得到（式 3.14）：



$$G(x) = 2J(x)^T F(x)$$
$$H(x) = 2J(x)^T J(x) + 2Q(x) \qquad \text{（式 3.14）}$$

其中：

$$Q(x) = \sum_{i=1}^{m} F_i(x) H_i(x)$$

随着 $x$ 接近最优解，$\|F(x)\|$的值趋近于零，$Q(x)$也趋近于零，因此，当$\|F(x)\|$在最优解处很小时，一个更有效的方法是使用 Gauss-Newton 搜索方向作为优化过程的搜索方向。下面将介绍两种最小二乘优化方法：Gauss-Newton 方法和 Levenberg-Marquardt 方法，MATLAB 优化工具箱中的 lsqnonlin 和 lsqcurvefit 函数使用了这两种方法求解非线性最小二乘优化问题。

## 3.4.1　Gauss-Newton 方法

### 1．算法描述

在 Gauss-Newton 方法中，每一次主迭代时得到一个搜索方向 $d_k$，从而得到最小二乘问题的一个可行解。

$$\min_{x \in R^n} \| J(x_k) d_k - F(x_k) \|_2^2$$

当 $Q(x)$可以忽略时，此时得到的搜索方向与 Newton 方法得到的搜索方向相同。$d_k$ 保证了在每次迭代时目标函数的值都在减小。但是，当（式 3.14）中的 $Q(x)$很大时，Gauss-Newton 方法很难能求得问题的解，在此种情况下，Levenberg-Marquardt 方法显得更有效。

### 2．算法的 MATLAB 实现

Gauss-Newton 方法是用前面求解无约束条件极值问题讨论过的多项式线性搜索策略来实现的。使用 Jacobian 矩阵 $J(x_k)$的 QR 分解，可以避免在求解线性最小二乘问题中等式条件恶化的问题。

该方法中含有一项鲁棒性检测技术，这种技术能够在步长小于门限值（在此实现中为 1e–5）或矩阵 $J(x_k)$的条件数小于 1e–10 时（矩阵条件数是指矩阵最大特征值与最小特征值的比值）能将算法过渡到 Levenberg-Marquardt 方法。

## 3.4.2　Levenberg-Marquardt 方法

### 1．算法描述

Levenberg-Marquardt 方法所使用的搜索方向是下列一组线性等式的解：

$$(J(x_k)^T J(x_k) + \lambda_k I) d_k = -J(x_k) F(x_k) \qquad \text{（式 3.15）}$$

其中：标量$\lambda_k$ 决定着搜索的方向和幅值大小，当$\lambda_k = 0$ 时，此时的搜索方向与 Gauss-

Newton 方法的搜索方向相同；当$\lambda_k$趋向于无穷大时，$d_k$趋向于零向量，从而得到一个最速下降方向，这也说明，只要$\lambda_k$足够大，就可以保证 $F(x_k+d_k)<F(x_k)$，因此$\lambda_k$可以解决影响 Gauss-Newton 方法效率的 $Q(x)$问题，保证函数值在迭代时是下降的。显然，Levenberg-Marquardt 方法的搜索方向介于 Gauss-Newton 方法的搜索方向和最速下降方向之间。

2．算法的.MATLAB 实现

Levenberg-Marquardt 方法实现的主要困难在于每一次迭代时如何控制$\lambda_k$的大小问题，这种控制可以使它对宽谱问题有效。在.MATLAB 中实现这一控制所采用的方法是使用线性预测平方总和 $F_p(x_k)$和最小 $F_k(x^*)$的三次内插值来估计 $F(x)$的相对非线性，在这种方法中，$\lambda_k$的大小在每一次迭代时都能确定。

线性预测平方总和按（式 3.16）进行计算：

$$\begin{aligned} F_p(x_k) &= J(x_{k-1})^T d_{k-1} + f(x) \\ f_p(x_k) &= F_p(x_k)^T F_p(x_k) \end{aligned} \quad \text{（式 3.16）}$$

其中：$f_k(x^*)$和步长参数$\alpha^*$可以通过三次内插值 $F(x_k)$和 $F(x_{k-1})$得到，$\alpha^*$是最小化的估计步长。如果 $f_p(x_k)$大于 $f_k(x^*)$，则$\lambda_k$减小，否则$\lambda_k$增加。$f_p(x_k)$和 $f_k(x^*)$之间差值的调整能力是 Gauss-Newton 方法有效性的一个度量，它决定了是使用最速下降搜索方向还是 Gauss-Newton 方法的搜索方向，其实现流程图如图 3-4 所示。

随着的$\lambda_k$更新，（式 3.15）的解就构成了一个搜索方向 $d_k$，然后 $d_k$增加一个单位迭代步长进行下一步搜索，如此类推，像无约束优化算法实现一样形成一个线性搜索过程，这个线性搜索过程保证在每次迭代时有 $f(x_{k+1})<f(x_k)$，因此这种方法也是一种下降方法。

这种实现方法在大量的非线性优化问题中得到了成功的应用，并被证明它比 Gauss-Newton 方法具有更好的鲁棒性，比无约束优化方法具有更好的迭代效率。在.MATLAB 优化工具箱中实现 Levenberg-Marquardt 方法的默认函数是 lsqnonlin。Gauss-Newton 方法可以通过设置 options 的参数 Levenberg-Marquardt 为 off 来实现。

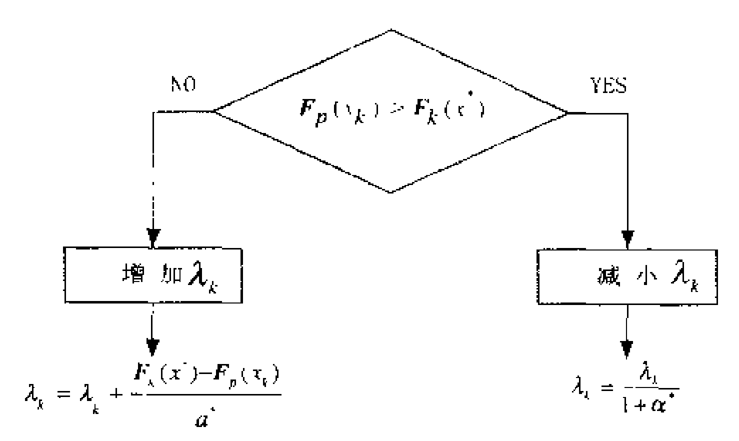



图 3-4 Levenberg-Marquardt 方法实现流程



在.MATLAB 的标准算法中，函数 lsqcurvefit 和 lsqnonlin 实现了使用线性搜索的 Levenberg-Marquardt 方法；另外，可以通过设置参数选项 Levenberg-Marquardt 为 off 来使用 Gauss-Newton 方法（特别是当残差$\|F(x)\|_2^2$

很小时，后者的求解速度更快）。其中默认的线性搜索算法是混合二次和三次多项式插值方法。

一般的最小二乘优化问题都可以用 MATLAB 优化工具箱提供的函数 lsqlin、lsqcurvefit 和 lsqnonlin 直接求解，但是对一些特殊情况还需自己编程来实现（见范例分析）。

### 3.4.3 范例分析

在（式 3.11）所表示的线性最小二乘优化问题中，当矩阵 ***C***、***A*** 或者 ***Aeq*** 是稀疏矩阵时，用 MATLAB 优化工具箱提供的函数可能得不到问题的解，此时必须自己编程来实现。

考虑以下形式的线性最小二乘问题：

$$\min_x f(x) = \min_x \frac{1}{2} \| C * x - d \|_2^2$$

$$约束为：|x - x_0| \leq r$$

其中：矩阵 ***C***, ***A*** 或者 ***Aeq*** 是稀疏矩阵。

**【程序清单】**

例程 3-13 是求解的 MATLAB 代码：rls.m。

例程 3-13

```
function [x,act]=rls(A,b,x0,r);
% 默认设置
if      nargin<3, fac=1000;x0=zeros(size(A,2),1);
elseif nargin<3, fac=x0;x0=zeros(size(A,2),1);
else     b=b-A*x0;
end;
x=x0;
aa=diag(A'*A);
% 处理零列
ind=find(aa>0);
n=length(ind);
if n==0, act=1; return; end;
A=A(:,ind);
gamma=b'*b;
if gamma==0, act=1; return: end;
if nargin<3, r=fac*sqrt(gamma./aa(ind));
else    r=r(ind);
end;
% 最小化 ||Az-b||^2   s.t. |z|≤r
prt=0;
[z,fct,ier]=minq(0,-A'*b,A'*A,-r,r,prt);
act=max(abs(z)./r);
```

```
x(ind)=x(ind)+z;
%测试程序 rlstest.m
n=20;
m=n+5;
sig=1e-7;
A=rand(m,n);
A(:,3)=A(:,n-5)+A(:,n-2)+sig*randn(m,1);
A(6,:)=A(7,:)+A(5,:)+sig*randn(1,n);
x0=randn(n,1);
b=A*x0+1e-2*randn(m,1);
% b=randn(m,1);
tic;x=A\b;res1=norm(b-A*x);,tim1=toc
tic;
[xx,act]=rls(A,b);
```

**【结果输出】**

```
xx =
        -1.1317
        0.0105
        -779.7136
        -1.9371
        -0.5372
        0.3088
        -0.5361
        -0.3172
        0.2087
        0.1622
        -1.0073
        0.7392
        1.3519
        0.6723
        779.1515
        0.3289
        -1.3683
        781.2266
        -0.5665
        -0.0313
act =
         0.5360
```

# 3.5 多目标优化算法及实现

本节将在第 2 章的基础上，详细介绍多目标优化问题的算法及实现。

### 3.5.1　多目标优化算法介绍

在实际工程实践中，人们所遇到的问题很少是带有几个固定约束条件的单个目标的优化问题，更多的时候遇到同时追求多个目标 $\boldsymbol{F}(x)=\{F_1(x), F_2(x), \cdots, F_n(x)\}$ 的最优化问题，这些目标必须按照某一准则赋予不同的权重。这些目标之间的相互重要性通常不知道，直到系统的性能指标达到最优或者对不同目标的权重有深刻的了解之后。随着目标数目的增加，权重变得更加复杂，不容易加以定量化。因此，解决多目标最优化问题的策略是能够将一个自然的实际问题用一个公式表达出来，从而能够解决它。

多目标优化问题涉及求解目标向量 $\boldsymbol{F}(x)$ 的最优化问题，有时还附带一定的约束条件，其标准形式如（式 3.17）所示

$$\min_{x\in R^n} \boldsymbol{F}(x)=\min_{x\in R^n}(f_1(x), f_2(x), \cdots, f_m(x)) \qquad \text{（式 3.17）}$$

且满足：

$$\begin{aligned} &\boldsymbol{G}_i(x)=0 \qquad i=1,\cdots,m_e \\ &\boldsymbol{G}_i(x)\leqslant 0 \qquad i=m_e+1,\cdots,m \\ &lb\leqslant x\leqslant ub \end{aligned}$$

### 3.5.2　目标逼近方法

利用目标逼近方法来求解多目标规化问题常常是指：已知一组对象集 $\boldsymbol{F}(x)=\{F_1(x), F_2(x), \cdots, Fm(x)\}$，期望的设计目标值 $\boldsymbol{F}^*=\{F_1^*, F_2^*, \cdots, Fm^*\}$，要求采用一定的方法使得设计出来的各个对象的相应指标逼近目标值；也就是说，都在期望的目标值附近（稍大或稍小），两者之间差值的大小可以通过设置一个权重系数 $w=(w_1, w_2, \cdots, w_m)$ 来控制。其标准形式如下：

$$\min_{x\in\Omega,\gamma\in R} \gamma$$

$$\text{使得：} F_i(x)-w_i*\gamma\leqslant \boldsymbol{F}_i, \qquad i=1,\cdots,m$$

在上述表达式中，引入了松弛变量 $w_i*\gamma$，它使得优化结果更能逼近设计目标值。

目标逼近方法为实际的优化问题提供了一种很方便的直观表达形式，并且可以采用标准的优化算法加以解决。目标逼近方法可以用几何图形来描述，用于二维问题（式 3.18）的目标逼近方法的几何意义如图 3-5 所示。

二维问题：

$$\min_{x\in\Omega,\gamma\in R} \gamma \qquad \text{（式3.18）}$$

$$使得:\quad \begin{aligned} F_1(x)-w_1*\gamma \le F_1^* \\ F_2(x)-w_2*\gamma \le F_2^* \end{aligned}$$

已知目标值为 $P=\{F_1^*,F_2^*\}$，权重向量 $\Lambda(\gamma)$ 定义了从目标点 $P$ 到可行函数空间的搜索方向。在优化过程中，$\gamma$ 的值不断变化，因而优化问题的可行区域不断变小，最终收敛到优化解 $F_{1s},F_{2s}$。

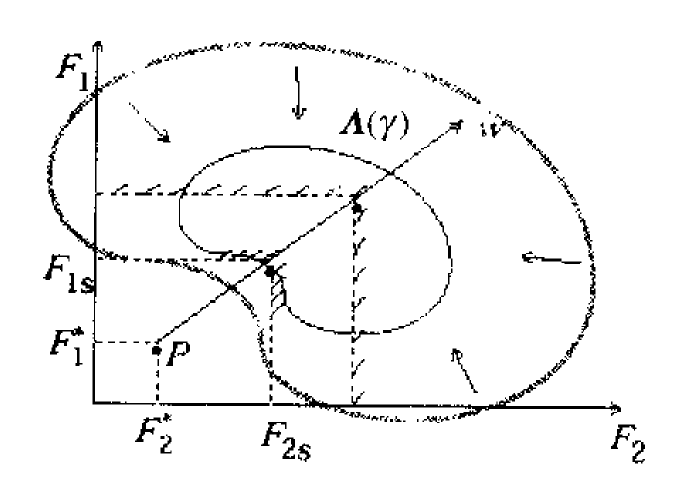



图 3-5 目标逼近方法的几何意义

### 3.5.3 目标逼近方法的改进

目标逼近方法的优点在于它还可以被看做一个非线性规划问题，这类问题本身的特点使得可以使用非线性优化算法来求解它。在前述的序列二次规划（SQP）算法中，为线性搜索过程选择一个指标函数是一件很难的事情，这是因为人们很难在改进目标函数值和减少不满足约束条件两者之间决定它们各自的相对重要性。在多目标规划方法中，将优化问题写成如（式 3.19）所示的最小化问题，我们便可能得到一个合适的指标函数。

$$\min_{x\in R^{\Omega}}\max_{i}(\Lambda_i) \qquad (式 3.19)$$

其中：$\Lambda_i=\dfrac{F_i(x)-F_i^*}{w_i}$

对（式 3.19）所示的目标逼近问题，使用序列二次规划（SQP）算法，可以把（式 3.20）作为指标函数：

$$\psi(x,\gamma)=\gamma+\sum_{i=1}^{\infty}r_i*\max\{0,F_i(x)-w_i\gamma-F_i^*\} \qquad (式 3.20)$$

但是以（式 3.20）作为线性搜索过程的基础时，会出现矛盾情况：虽然$\psi(x,\gamma)$在给定的搜索方向上减少一个步长，但函数$\max \Lambda_i$的值反而会增加。有一种解决方法就是设置$\psi(x)$函数为：

$$\psi(x)=\sum_{i=1}^{m}\begin{cases} r_i * \max\{0, F_i(x)-w_i\gamma-F_i^*\}, w_i=0 \\ \max\limits_i \Lambda_i, \quad \text{其他} \end{cases}$$

在.MATLAB 优化工具箱中，函数 fgoalattain 实现了上述改进，使得多目标规划算法更具有鲁棒性。

### 3.5.4 范例分析

如图 3-6 所示，这个模型中含有一个非线性处理单元，它的模型见图中对应的部分。这个单元是一个输入激励受限的三阶欠阻尼模型，限制函数如图 3-7 所示，斜率为 0.8。这个开环系统的阶跃响应如图 3-8 所示。

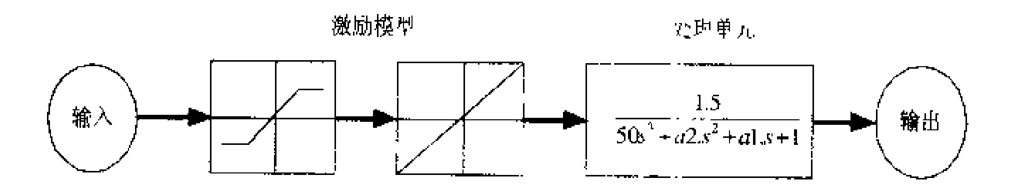



图 3-6 开环系统

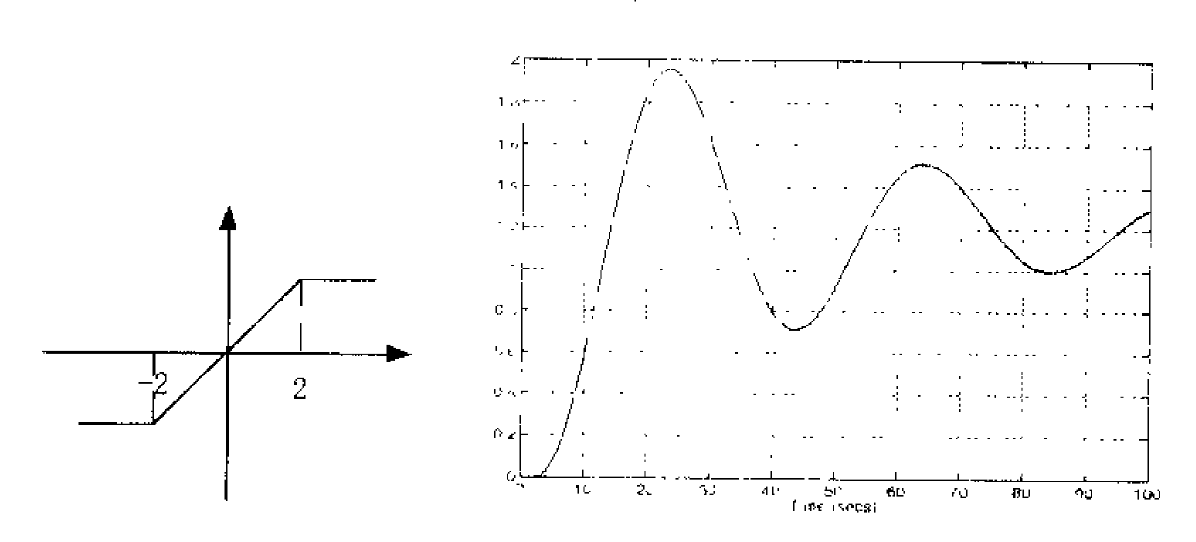



图 3-7 限制函数

图 3-8 开环系统的阶跃响应

要求设计一个反馈控制规律，使得系统的输出能跟踪单位阶跃信号。

【分析】

如图 3-9 所示，采用 PID 控制。原系统中的激励模型和非线性处理单元被一个方框图代替，作为闭环系统的一部分。解决这个问题的一种方法就是使得系统的输出和输入的差值最小，参数就是 PID 参数。如果仅在某一个时间点上使得差值最小，这就是一个单目标优化问题；但是此处要求在一段时间上（第 1～100 个时间步长），从而得到一个多目标优化问题，优化变量就是 PID 控制参数 $K_p$、$K_i$、$K_d$；另一种方法就是使用 fminimax 函数，此时不是使得系统的输出和输入的差值最小，而是使得在时间 0～100 内的输出的最大值最小。本例使用 MATLAB 优化工具箱的 lsqnonlin 和 fminimax 函数分别来控制系统输出跟踪输入。

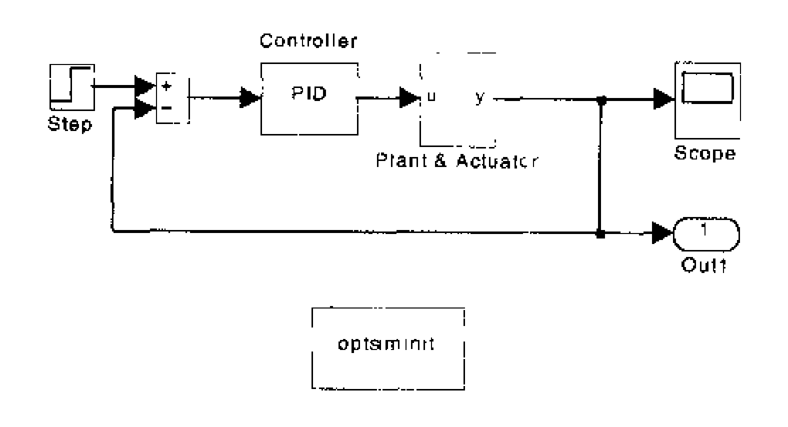



图 3-9　PID 控制

【程序清单】

[方法一]

例程 3-14 是求解的 MATLAB 代码。

例程 3-14

```
%编写返回差值的.m 文件
function F = tracklsq(pid,a1,a2)
%设置 PID 参数
Kp = pid(1);
Ki = pid(2);
Kd = pid(3);
%返回输出值
opt = simset('solver','ode5','SrcWorkspace','Current');
[tout,xout,yout] = sim('optsim',[0 100],opt);
%返回差值
F = yout-1;

%调用优化函数
%调入闭环系统模型
```

```
optsim
%设置 PID 参数初值
pid0=[0.63,0.0503,1.9688];
%初始化非线性处理单元的变量
a1=3;
a2=33;
options=optimset('largescale','off','display',...
    'iter','tolx', 0.001,'tolfun',0.001);
pid=lsqnonlin('tracklsq',pid0,[],[],options,a1,a2)
```

【结果输出】

经过 73 次估计目标函数值后，得出 PID 参数的最优值：

```
pid =
        2.9108      0.1333     12.8161
```

迭代过程参数如表 3-2 所示。

表 3-2　迭代过程参数

| 迭代次数 | 函数值估计次数 | 偏差大小 | 步长大小 | Directional derivative | Lambda |
|---|---|---|---|---|---|
| 1 | 3 | 8.66531 | 1 | -3.38 | 4.1123 |
| 2 | 10 | 6.78831 | 1 | -0.0633 | 3.3355 |
| 3 | 19 | 5.99203 | 5.5 | -0.0336 | 0.28612 |
| 4 | 28 | 3.73992 | 5.78 | -0.0213 | 0.0227966 |
| 5 | 36 | 3.51795 | 1.25 | 0.0222 | 0.0733258 |
| 6 | 33 | 3.5115 | 0.58 | -0.00633 | 0.03335 |
| 7 | 51 | 3.39355 | 2.99 | 0.000688 | 0.017225 |
| 8 | 58 | 3.3836 | 0.915 | 0.00203 | 0.0180998 |
| 9 | 66 | 3.37723 | 1.22 | 0.000835 | 0.00903992 |
| 10 | 73 | 3.37305 | 0.801 | -0.00072 | 0.0113309 |

闭环系统阶跃响应如图 3-10 所示。

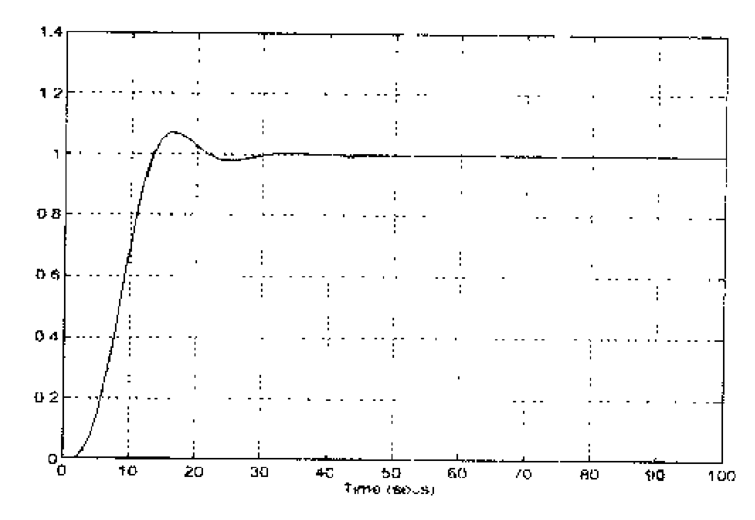



图 3-10　闭环系统阶跃响应

[方法二]

例程 3-15 是求解的 MATLAB 代码。

例程 3-15

```
%编写计算目标函数的.m 文件
function F = trackmmobj(pid,a1,a2)
Kp = pid(1);
Ki = pid(2);
Kd = pid(3);
% 计算函数值
opt = simset('solver','ode5','SrcWorkspace','Current');
[tout,xout,yout] = sim('optsim',[0 100],opt);
F = yout;
assignin('base','F_TRACKMMOBJ',F);

%编写计算非线性约束的.m 文件
function [c,ceq] = trackmmcon(pid,a1,a2)
F = evalin('base','F_TRACKMMOBJ');
% 计算约束
c = -F(20:100)+.95;
ceq = [ ];

%调用约束优化函数
optsim
pid0 = [0.63 0.0503 1.9688]
a1 = 3; a2 = 33;
options = optimset('Display','iter',...
                   'TolX',0.001,'TolFun',0.001);
pid = fminimax(@trackmmobj,pid0,[],[],[],[],[],[],...
              'trackmmcon',options,a1,a2)
% Put variables back in the base workspace
Kp = pid(1); Ki = pid(2); Kd = pid(3);
```

**【结果输出】**

经过 29 次估计目标函数值后，得出 PID 参数的最优值：

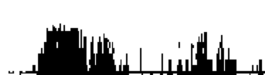
```
pid =
       0.5893      0.0605      5.5295
```

迭代过程如表 3-3 所示。

表 3-3 迭代过程

| 迭代次数 | 函数值估计次数 | 偏差大小 | 步长大小 | Directional erivative |
|---|---|---|---|---|
| 1 | 5 | 1.12 | 1 | 1.18 |
| 2 | 11 | 1.263 | 1 | -0.172 |
| 3 | 17 | 1.055 | 1 | -0.0128 |





（续表）

| 迭代次数 | 函数值估计次数 | 偏差大小 | 步长大小 | Directional erivative |
|---|---|---|---|---|
| 3 | 23 | 1.003 | 1 | 3.38e-005 |
| 5 | 29 | 0.9997 | 1 | -1.36e-006 |

Active Constraints:

1

13

182

闭环系统阶跃响应如图 3-11 所示。

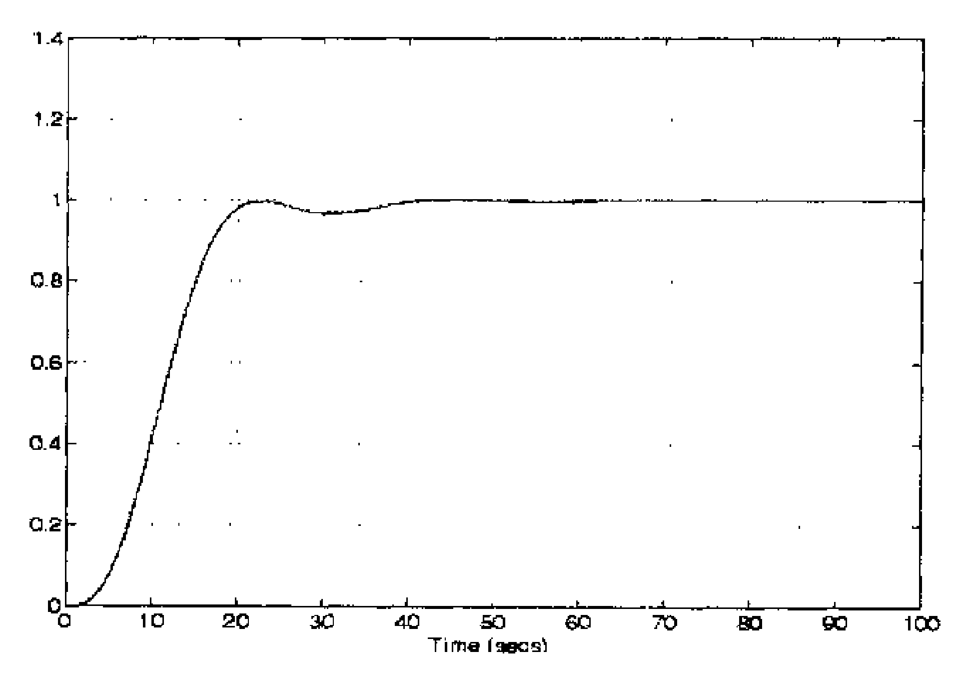



图 3-11　闭环系统阶跃响应

显然，调用 fminimax 函数与调用 lsqnonlin 函数的结果不相同。



# 第4章 工程优化算法及其MATLAB实现（二）——大规模算法

大规模系统优化问题一般是指含有稀疏结构的优化问题，而大规模系统优化算法一般采用迭代的一系列近似线性系统去逼近原系统。在每次迭代中，只需要解一个线性系统问题；另一方面，利用 MATLAB 6.5 可以处理稀疏矩阵和许多稀疏线性问题，这些线性系统问题可以很方便地通过直接方法或者迭代方法来求解。

总地来讲，大规模系统优化算法使用额外的迭代信息可以解决具有稀疏结构的优化问题。为了解决问题更有效，有些问题的求解条件是很严格的，甚至要求必需的附加信息。

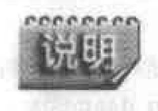
如表 4-1 所示，并非所有的大规模系统优化问题都能直接使用 MATLAB 中大规模系统优化算法来解决，表中第二列表示了可以解决的大规模系统优化问题，第三列表示解决此类问题需要的附加条件，第四列表示具有什么样特征的大规模系统使用此优化算法可以提高效率。

**表 4-1 MATLAB 可以解决的大规模系统优化问题**

| 函数 | 数学模型 | 必需的附加条件 | 适用系统 |
|---|---|---|---|
| fminunc | $\min\limits_{x} \quad f(x)$ | $f(x)$ 的梯度 | 提供了海色矩阵的稀疏结构；<br>海色矩阵是稀疏的 |
| fmincon | $\min\limits_{x} \quad f(x)$<br>s.t. $l \le x \le u$ 且 $l < u$<br>$\min\limits_{x} \quad f(x)$<br>s.t. $Aeq * x = Beq$ 且 $Beq$ 是 $m \times n$ 矩阵，$m \le n$ | $f(x)$ 的梯度 | 提供了海色矩阵的稀疏结构；<br>海色矩阵是稀疏矩阵；<br>$Aeq$ 是稀疏矩阵 |
| lsqnonlin | $\min\limits_{x} \quad \frac{1}{2}\|F(x)\|_2^2 = \frac{1}{2}\sum_i F_i(x)^2$<br>$\min\limits_{x} \quad \frac{1}{2}\|F(x)\|_2^2 = \frac{1}{2}\sum_i F_i(x)^2$<br>s.t. $l \le x \le u$ 且 $l < u$<br>$F(x)$ 的等式数目大于等于变量的数目 | 无 | 提供了雅可比矩阵的稀疏结构；<br>雅可比矩阵是稀疏的 |

（续表）

| 函　数 | 数学模型 | 必需的附加条件 | 适用系统 |
| --- | --- | --- | --- |
| lsqcurvefit | $\min\limits_{x} \frac{1}{2}\lVert F(x,xdata)-ydata\rVert_2^2$<br>$\min\limits_{x} \frac{1}{2}\lVert F(x,xdata)-ydata\rVert_2^2$<br>s.t. $l \leqslant x \leqslant u$ 且 $l<u$<br>$F(x,xdata)$的等式数目大于等于变量的数目 | 无 | 提供了雅可比矩阵的稀疏结构；<br>雅可比矩阵是稀疏的；<br>$Aeq$ 是稀疏矩阵 |
| fsolve | $F(x)=0$，$F(x)$有同样数目的变量和等式 | 无 | 提供了雅可比矩阵的稀疏结构；<br>雅可比矩阵是稀疏的 |
| lsqlin | $\min\limits_{x} \lVert Cx-d\rVert_2^2$<br>s.t. $l \leqslant x \leqslant u$ 且 $l<u$<br>$C$ 是$m \times n$矩阵，且$m \leqslant n$ | 无 | $C$ 是稀疏矩阵 |
| linprog | $\min\limits_{x} f^T x$<br>s.t. $A*x \leqslant b$，$Aeq*x=Beq$，<br>其中$l \leqslant x \leqslant u$ | 无 | $A$ 和 $Aeq$ 是稀疏矩阵 |
| quadprog | $\min\limits_{x} \frac{1}{2}x^T H x+f^T x$<br>s.t. $l \leqslant x \leqslant u$ 且 $l<u$；<br>$\min\limits_{x} \frac{1}{2}x^T H x+f^T x$；<br>s.t. $Aeq*x=Beq$ 且 $Beq$ 是 $m \times n$ 矩阵，$m \leqslant n$ | 无 | $C$ 是稀疏矩阵<br>$Aeq$ 是稀疏矩阵 |




当然，对于一般系统的优化问题也可以使用大规模系统优化算法来解决，但这不能提高多大的效率，因而显得没有这个必要。建议对于一般中小规模的系统，还是使用一般中小系统的标准优化算法。

本章主要内容：

- 工程优化算法的基本原理
- 非线性等式求解算法及实现
- 非线性最小二乘问题
- 非线性最小化问题
- 二次规划问题
- 线性最小二乘问题
- 大规模线性优化问题



# 4.1　工程优化算法的基本原理

本节将讨论工程优化算法的基本原理——信赖域法和预处理共轭梯度法。而大规模系统优化算法一般采用迭代的一系列近似线性系统去逼近原系统，每次迭代只需要解一个线性系统问题，所以本节还要研究线性约束问题。

## 4.1.1　信赖域法

在大规模优化算法中，许多优化函数的原理是基于信赖域法的，因此我们必须理解信赖域法的基本原理。信赖域法的基本原理是：首先指定一个最大步长，然后用带约束的二次模型来确定惟一的方向与大小。这个最大步长提供了一个使该二次模型成为 $f(x)$ 的可信赖域近似模型的区域，信赖域法因此得名。许多研究者认为，这种方法是一个能够提供非常有效并具有十分精致的整体收敛性质的最优化算法。

## 4.1.2　预处理共轭梯度法（PCG 法）

解决具有线性等式 $\boldsymbol{H}p=-g$ 的大规模对称正定系统的最常用的方法是预处理共轭梯度法（PCG）。这种迭代方法要求能够计算矩阵向量的乘积 $\boldsymbol{H}\cdot\boldsymbol{v}$，其中 $\boldsymbol{v}$ 是任意一个向量。对称正定矩阵 $\boldsymbol{M}$ 是 $\boldsymbol{H}$ 先决矩阵，即 $\boldsymbol{M}=\boldsymbol{C}^2$，$\boldsymbol{C}^{-1}\boldsymbol{H}\boldsymbol{C}^{-1}$ 是奇异矩阵或是满秩的。

优化工具箱采用PCG算法的步骤如下所示。

（1）初始化

```
r = -g;
p = zeros(n,1);
```

（2）预处理

```
z = M\r;
inner1 = r'*z;
inner2 = 0;
d = z;
```

（3）共轭梯度迭代

```
for k = 1:kmax
if k > 1
beta = inner1/inner2;
d = z + beta*d;
end
w = H*d; denom = d'*w;
if denom <= 0
p = d/norm(d); %负或零曲率方向
```

```
break;    %如果发现负或零曲率方向，停止；
else
alpha = inner1/denom;
p = p + alpha*d;
r = r - alpha*w;
end
z = M\r;
if norm(z) <= tol    %如果在容许误差内 Hp=-g 则退出
break
end
inner2 = inner1;
inner1 = r'*z;
end
```

在求最小化问题时，可以假设海色矩阵是对称的，$\boldsymbol{H}$ 是正定的且仅在严格极小点的邻域内。PCG 的输出方向 $p$ 要么是负曲率方向，要么是逼近牛顿法 $\boldsymbol{H}p=-g$ 的解方向，它用来寻找二维的子空间，以便使用信赖域法来解决极值问题。PCG 算法仅在发现负或零曲率方向时方可退出，即 $\boldsymbol{d}^T\boldsymbol{H}\boldsymbol{d}\leqslant 0$。

### 4.1.3 线性约束问题

虽然线性约束条件比无约束问题更加复杂，但在 MATLAB 中大规模系统优化算法采用了高效的方法：

- 在有线性约束时使用投射法；
- 在有简单两端约束时使用反射法。

对于线性约束问题，一般可写成这种形式

$$\begin{aligned}&\min f(\boldsymbol{x})\\&s.t\quad \boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}\end{aligned}\qquad（式 4.1）$$

这里 $\boldsymbol{A}$ 是 $m$ 行 $n$ 列矩阵。优化工具箱先采用 LU 分解方法去除线性相关性，假设矩阵 $\boldsymbol{A}$ 的秩为 $m$，解决此问题的高效算法与普通的算法的区别在于两个重要方法：第一，采用稀疏最小二乘算法，计算初始可行点；第二，使用 RPCG 法来代替 PCG 方法，目的是计算逼近减小的牛顿步长（或负曲率方向）。

对于两端约束的一般形式为

$$\begin{aligned}&\min f(\boldsymbol{x})\\&s.t\quad \boldsymbol{l}\leqslant \boldsymbol{x}\leqslant \boldsymbol{u}\end{aligned}\qquad（式 4.2）$$

其中 $\boldsymbol{l}$ 和 $\boldsymbol{u}$ 为向量，它们的某个元素可以为 $\infty$，此方法可产生严格的可行点序列。为了使算法收敛，两个技术被用来保证可行性：第一，可变尺度牛顿步长代替了无约束牛顿步长（可以定义两个子空间）；第二，反射法用来增加步长的长度。



通过检查（式 4.3）的 K-T 必要条件，可以得到可变尺度牛顿步长。

$$\left(D(x)\right)^{-2} g = 0 \qquad （式 4.3）$$

其中

$$D(x) = diag\left(\left|v(x)\right|^{-\frac{1}{2}}\right)$$

变量 $v$ 是这样定义的（$l \leqslant i \leqslant u$）：

如果 $g_i < 0$ 和 $u_i < \infty$ 则 $v_i = x_i - u_i$

如果 $g_i \geqslant 0$ 和 $l_i > -\infty$ 则 $v_i = x_i - l_i$

如果 $g_i < 0$ 和 $u_i = \infty$ 则 $v_i = -l_i$

如果 $g_i \geqslant 0$ 和 $l_i < -\infty$ 则 $v_i = l_i$

（式 4.3）并不是处处可微的，例如在 $v_i = 0$ 时是不可微的。为了保证严格可行性，必须避免这个点，即限制 $l < x < u$。

对于（式 4.3），可变尺度牛顿步长定义为此线性系统的解，即：

$$\hat{M}Ds^N = -\hat{g} \qquad （式 4.4）$$

其中

$$\hat{g} = D^{-1}g = diag\left(|v|^{\frac{1}{2}}\right)g \qquad （式 4.5）$$

和

$$\hat{M} = D^{-1}HD^{-1} + diag(g)J^v \qquad （式 4.6）$$

其中 $J^v$ 是 $|v|$ 的 Jacobian 矩阵，其对角线每个元素为 0、-1、1。如果 $l$ 和 $u$ 的每个元素都是有限的，则 $J^v = diag(sign(g))$。在 $g_i = 0$ 点处，$v_i$ 可能是不可微的，此时可令 $J^v_{ii} = 0$，原因是对此元素，$v_i$ 的取值是不重要的；另外，在此点还可能是不连续的，但是 $|v_i| \cdot g_i$ 却是连续的。

反射法用来增加步长。一个反射步长可以这样定义：假设第 $i$ 个约束的搜索方向为 $p$，

则反射步长 $p^R = p$，除了在第 $i$ 个元素处，$p_i^R = -p_i$

## 4.2 非线性等式求解算法及实现

本节我们研究非线性等式求解算法及其 MATLAB 实现，这个问题对于我们分析大规模系统优化算法很有启发和帮助作用。

### 4.2.1 非线性等式求解算法简介

对于线性等式这一类问题，可以用线性代数的方法来判断并求出此类问题的完全解；而对于复杂的非线性等式问题，必须用逼近的方法来求解，即使用迭代的点序列使目标值逐渐逼近零点。逼近的方法有牛顿法、共轭梯度法等。

### 4.2.2 范例分析

**1．使用 Jacobian 矩阵求解非线性等式**

求解 $\boldsymbol{x}$，使 $\boldsymbol{F}(\boldsymbol{x})=0$，其中 $j \leqslant k-1$，且有

$$
\begin{cases}
F(1) = 3x_1^2 - 2x_1^2 - 2x_2 + 1 \\
\vdots \\
F(i) = 3x_i^2 - 2x_i^2 - x_{i-1} - 2x_{i+1} + 1 \\
\vdots \\
F(n) = 3x_n - 2x_n^2 - x_{i-1} + 1
\end{cases}
$$

$$n = 1000$$

**【分析】**

易知，$\boldsymbol{F}(\boldsymbol{x})$ 的 Jacobian 矩阵是稀疏的，因而可以使用 fsolve 的大规模算法来解决。

**【程序清单】**

例程 4-1 是求解的 MATLAB 代码。

例程 4-1

```
%写一个计算目标函数值和 Jacobian 矩阵的 M-file nlsf1.m
function [F,J] = nlsf1(x);
% 求目标函数的值
n = length(x);
```

```
F = zeros(n,1);
i = 2:(n-1);
F(i) = (3-2*x(i)).*x(i)-x(i-1)-2*x(i+1)1+ 1;
F(n) = (3-2*x(n)).*x(n)-x(n-1) + 1;
F(1) = (3-2*x(1)).*x(1)-2*x(2) + 1;
% 如果 nargout > 1，计算雅克比矩阵 J 的值
if nargout > 1
      d = -4*x + 3*ones(n,1); D = sparse(1:n,1:n,d,n,n);
c = -2*ones(n-1,1); C = sparse(1:n-1,2:n,c,n,n);
      e = -ones(n-1,1); E = sparse(2:n,1:n-1,e,n,n);
      J = C + D + E;
End

%调用函数 fsolve 来解此等式
xstart = -ones(1000,1);
fun = @nlsf1;
options = optimset('Display','iter','Jacobian','on');
[x,fval,exitflag,output] = fsolve(fun,xstart,options);
```

**【说明】**

xstart是选择的初始点，由于在MATLAB中，大规模算法是默认的，所以在使用函数optimset设置选项时，不需要再特别说明；fsolve需要使用Jacobian矩阵的信息，所以在设置选项时，设置'Jacobian'参数为'on'；运算输出包括每一次迭代的结果。

**【结果输出】**

| Iterations | Func-count CG-Iteration | $f(x)$ | Norm of step | First-order Optimality | CG of Iterations |
|---|---|---|---|---|---|
| 1 | 2 | 1011 | 1 | 19 | 0 |
| 2 | 3 | 16.1942 | 7.91898 | 2.35 | 3 |
| 3 | 4 | 0.0228027 | 1.33+2 | 0.291 | 3 |
| 4 | 5 | 0.000103359 | 0.0433329 | 0.0201 | 4 |
| 5 | 6 | 7.3792e-007 | 0.0022606 | 0.000946 | 4 |
| 6 | 7 | 4.02299e-010 | 0.000268381 | 4.12e-005 | 5 |

可以看出，最后得到的值已在容许误差内。

**【其他解法】**

对于近似线性系统，它迭代的算法可以采用先决共轭梯度方法。由于PrecondBandWidth在option中默认为0，因而对角先决方法被使用（PrecondBandWidth是指先决矩阵的带宽，带宽为0意味着矩阵中只有一条对角线）。

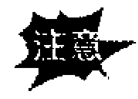
一阶优化，可以实现快速线性收敛。共轭梯度（CG）迭代的次数要求每一次主要的迭代步数是最低的，例如对于一个1 000维的问题至多需要5步，意味着线性系统的等式求解并不困难。

另外，也可以通过设置参数PrecondBandWidth的值来选择有界的先决，例如如果你想使用三角对角线矩阵，可以设置PrecondBandWidth的值为1。

```
options = optimset('Display','iter','Jacobian','on','PrecondBandWidth',1) ;
[x,fval,exitflag,output] = fsolve(fun,xstart,options);
```

【结果输出】

| Iterations | Func-count | f(x) | Norm of step | First-order optimality | CG-Iteration |
|---|---|---|---|---|---|
| 1 | 2 | 1011 | 1 | 19 | 0 |
| 2 | 3 | 16.0839 | 7.92496 | 1.92 | 1 |
| 3 | 4 | 0.0458181 | 1.3279 | 0.579 | 1 |
| 4 | 5 | 0.000101184 | 0.0631894 | 0.0203 | 2 |
| 5 | 6 | 3.16615e-007 | 0.00273698 | 0.00079 | 2 |
| 6 | 7 | 9.72481e-010 | 0.00018111 | 4.82e-005 | 2 |



结果比较表明，尽管迭代的次数一样，但是共轭梯度迭代的次数已经下降了，所以每一步的工作量减少了。

2．用稀疏结构的雅克比矩阵求解非线性等式

在前面的例子中，函数*nlsf1*在计算矢量函数的同时也计算了稀疏矩阵——Jacobian矩阵***J***。但是如果Jacobian矩阵计算不出，怎么办呢？在函数fsolve、lsqnonlin和lsqcurvefit中将默认采用有限微分矩阵来代替Jacobian矩阵。

【分析】

为了使代替Jacobian矩阵的有限微分矩阵有效，必须提供Jacobian矩阵的稀疏模型，这可以通过在options中设置JacobPattern参数来实现。也就是说，提供一个非零元素的个数与Jacobian矩阵相等的稀疏矩阵Jstr。实际上，Jstr的非零集对应于Jacobian矩阵相应非零集的父集，总地来说，随着Jstr非零元素的增加，计算稀疏有限微分矩阵的开销也会增加。

对于大型的问题，提供稀疏矩阵模型可以大大减少所需计算有限微分矩阵的时间。例如前一个例子，如果稀疏矩阵未提供，有限微分将计算在Jacobian矩阵中的1 000×1 000个元素；而如果稀疏矩阵提供了，有限微分只需计算其中的2 998个非零值，将远远小于需要的1 000 000个，可以看出，对于大型问题，必须提供稀疏矩阵模型，否则计算机在计算有限微分时容易溢出。当然对于大部分小型的问题，提供稀疏矩阵就没有必要了。

【程序清单】

事先计算稀疏矩阵 Jstr 存储在一个文件中，对于本例为 nlsdat1.mat。对函数 *nlsf1a* 使用函数 fsolve，使用稀疏有限微分矩阵估计 Jacobian 矩阵。

例程 4-2 是求解的 MATLAB 代码。

例程 4-2

```
%写一个计算目标函数值和 Jacobian 矩阵的 M-file nlsf1.m
```

```
function F = nlsf1a(x);
%计算目标向量的值
n = length(x);
F = zeros(n,1);
i = 2:(n-1);
F(i) = (3-2*x(i)).*x(i)-x(i-1)-2*x(i+1) + 1;
F(n) = (3-2*x(n)).*x(n)-x(n-1) + 1;
F(1) = (3-2*x(1)).*x(1)-2*x(2) + 1;

%调用函数来解此等式
xstart = -ones(1000,1);
fun = @nlsf1a;
load nlsdat1 % Get Jstr
options = optimset('Display','iter', 'JacobPattern',Jstr, 'PrecondBandWidth',1);
[x,fval,exitflag,output] = fsolve(fun,xstart,options);
```

【结果输出】

| Iterations | Func-count | f(x) | Norm of step | First-order optimality | CG-Iteration |
|---|---|---|---|---|---|
| 1 | 6 | 1011 | 1 | 19 | 0 |
| 2 | 11 | 16.0839 | 7.92496 | 1.92 | 1 |
| 3 | 16 | 0.0458181 | 1.3279 | 0.579 | 1 |
| 4 | 21 | 0.000101184 | 0.0631898 | 0.0203 | 2 |
| 5 | 26 | 3.16615e-007 | 0.00273698 | 0.00079 | 2 |
| 6 | 31 | 9.72482e-010 | 0.00018111 | 4.82e-005 | 2 |

【其他解法】

另外，可以选择直接线性解法（稀疏矩阵QR分解法，即我们设置参数PrecondBandWidth值为无穷大）。直接线性解法将会代替先决共轭矩阵解法。

```
xstart = -ones(1000,1);
fun = @nlsf1a;
load nlsdat1 % Get Jstr
options = ptimset('Display','iter','JacobPattern',Jstr,'PrecondBandWidth',inf);
[x,fval,exitflag,output] = fsolve(fun,xstart,options);
```

【结果输出】

| Iterations | Func-count | f(x) | Norm of step | First-order optimality | CG-Iteration |
|---|---|---|---|---|---|
| 1 | 6 | 1011 | 1 | 19 | 0 |
| 2 | 11 | 14.9018 | 7.92421 | 1.89 | 1 |
| 3 | 16 | 0.0128163 | 1.32542 | 0.0746 | 1 |
| 4 | 21 | 1.73538e-008 | 0.0397925 | 0.000196 | 1 |
| 5 | 26 | 1.13169e-018 | 4.55544e-005 | 2.76e-009 | 1 |



注意　从结果比较可以看出，在采用直接方法时，CG 的值始终为 1。

## 4.3　非线性最小二乘问题

本节我们研究大规模系统优化的非线性最小二乘问题。

### 4.3.1　非线性最小二乘问题简介

对于最小二乘问题

$$f(x)=\frac{1}{2}\sum_i f_i^2(x)=\frac{1}{2}\|F(x)\|_2^2 \qquad \text{（式 4.7）}$$

其中 $F(x)$ 为向量函数，它的第 $i$ 个元素为 $f_i(x)$。它的基本解决方法是使用信赖域法，但是，由于它的非线性二次结构可以采用 Guass-Newton 方向，使之解决的效率大大提高，即

$$\min\|Js+F\|_2^2 \qquad \text{（式 4.8）}$$

其中 $J$ 是的 $F(x)$ 的 Jacobian 矩阵，用来定义两维子空间 $s$，而不是使用 $f_i(x)$ 的二阶导数。

在每次迭代中，使用 PCG 方法来解正规方程，即

$$J^T Js=-J^T F$$

### 4.3.2　范例分析

大规模优化算法ilsqnonlin、lsqcurvefit和fsolve也可以用于小或中规模系统，这时不需要提供Jacobian稀疏矩阵模型（这也适用于大规模算法fmincon 或 fminunc，不需要提供Hessian稀疏矩阵模型）。当然，判断某系统是否为大规模系统或是小或中规模系统，取决于计算机的处理能力。

假设某问题的约束条件有$m$个等式，$n$个未知数。如果令$J$ = sparse(ones($m$,$n$))，在计算时引起了内存溢出的问题，你就可以认为这个问题是一个大规模问题。

下面举一个例子，此问题有 10 个因子和 2 个未知数，求 $x_1$，$x_2$。

目标函数：　$\min\sum_{k=1}^{10}\left(2+2k-e^{kx_1}-e^{kx_2}\right)^2$

【分析】

因为非线性最小二乘优化函数lsqnonlin需要向量形式的目标函数，因此首先我们把目标函数重写为向量形式：

$$F_k(x)=2+2k-e^{kx_1}-e^{kx_2} \quad k=1,2,\cdots,10$$

【程序清单】

例程 4-3 是求解的 MATLAB 代码.

例程 4-3

```
%写一个计算目标函数值 M-file myfun.m
function F = myfun(x)
k = 1:10;
F = 2 + 2*k-exp(k*x(1))-exp(k*x(2));

%调用函数 lsqnonlin 来解此等式
x0 = [0.3 0.4] %初始点
[x,resnorm] = lsqnonlin(@myfun,x0) % 使用优化函数
```

【说明】

在本例myfun.m 中没有计算Jacobian矩阵，也没有在options中提供Jacobian矩阵模型，sqnonlin 将采用默认Jacobian矩阵模型Jstr = sparse(ones(10,2))，这是Lsqnonlin默认的设置.需要提醒读者的是，在options中Jacobian参数默认为'off'。

当使用有限微分算法时，它先检测到Jstrsparse(ones(10,2))实际上是一个稠密矩阵，由于把它当成稀疏矩阵并不会加快计算速度，于是它将把Jstr当成一个稠密矩阵ones(10,2)进行优化计算。

【结果输出】

```
x =
      0.2578 0 2578
      resnorm =124.3622
```

虽然现在的大部分计算机系统不用大规模算法也可以处理比较大的问题（这些问题一般少于 100 个等式和变量），但是利用 Jacobian 或 Hessian 稀疏结构，并且采用大规模算法可以大大提高处理速度。

# 4.4 非线性最小化问题

本节我们研究大规模系统优化的非线性最小化问题

## 4.4.1 非线性最小化问题简介

非线性最小化问题包括无约束的极小化问题和有约束的极小化问题. 预处理共轭梯度

法对此类问题是非常有效的，原因是共轭梯度法不但与拟牛顿法一样只利用函数的梯度信息，而且不用存贮二阶导数矩阵。

求解约束优化问题的思路主要分为两大类：一是直接对目标函数采用搜索法在可行方向求出最优解；二是对目标函数进行转换，化为更易求解的问题来求原优化问题的最优解。

下面我们对它的几个典型问题分别进行举例说明

### 4.4.2 范例分析

解决非线性最小化问题，一是采用明确的三阶海色矩阵来求解；二是用有限差分规则提供海色矩阵稀疏结构来求解。

问题是：

$$\min f(x)=\sum_{i=1}^{n-1}\left[\left(x_i^2\right)^{\left(x_{i+1}^2+1\right)}+\left(x_{i+1}^2\right)^{\left(x_i^2+1\right)}\right],\quad n=1\ 000$$

我们先使用它的函数梯度和海色矩阵来对此非线性最小化问题进行求解。

#### 1. 使用三阶海色矩阵求解

**【程序清单】**

例程 4-4 是求解的 MATLAB 代码。

例程 4-4

```
%写一个函数，它计算目标函数值、目标函数的梯度和稀疏三阶海色矩阵
function [f,g,H] = brownfgH(x)

% 计算目标函数
n=length(x); y=zeros(n,1);
i=1:(n-1);
y(i)=(x(i).^2).^(x(i+1).^2+1)+(x(i+1).^2).^(x(i).^2+1);
f=sum(y);
% 计算函数梯度
if nargout > 1
      i=1:(n-1); g = zeros(n,1);
      g(i)= 2*(x(i+1).^2+1).*x(i).*((x(i).^2).^(x(i+1).^2))+...
              2*x(i).*((x(i+1).^2).^(x(i).^2+1)).*log(x(i+1).^2);
      g(i+1)=g(i+1)+ 2*x(i+1).*((x(i).^2).^(x(i+1).^2+1)).*log(x(i).^2)+...
                2*(x(i).^2+1).*x(i+1).*((x(i+1).^2).^(x(i).^2));
end
% 计算稀疏对称海色矩阵
if nargout > 2
v=zeros(n,1);
i=1:(n-1);
v(i)=2*(x(i+1).^2+1).*((x(i).^2).^(x(i+1).^2))+4*(x(i+1).^2+1).*(x(i+1).^2).*(x(i).^2).* ...
((x(i).^2).^((x(i+1).^2)-1))+ 2*((x(i+1).^2).^(x(i).^2+1)).*(log(x(i+1).^2));
```

```
v(i)=v(i)+4*(x(i).^2).*((x(i+1).^2).^(x(i).^2+1)) *((log(x(i+1).^2)).^2);
v(i+1)=v(i+1)+2*(x(i).^2).^(x(i+1).^2+1).*(log(x(i).^2))+...
4*(x(i+1).^2).*((x(i).^2).^(x(i+1).^2+1)).*((log(x(i).^2)).^2)+2*(x(i).^2+1).*((x(i+1).^2).^(x(i).^2));
v(i+1)=v(i+1)+4*(x(i).^2+1).*(x(i+1).^2).*(x(i).^2).*((x(i+1).^2).^(x(i).^2-1));
 v0=v;
 v=zeros(n-1,1);
 v(i)=4*x(i+1).*x(i).*((x(i).^2).^(x(i+1).^2))+4*x(i+1).*...
(x(i+1).^2+1).*x(i).*((x(i).^2).^(x(i+1).^2)).*log(x(i).^2);
 v(i)=v(i)+4*x(i+1).*x(i).*((x(i+1).^2).^(x(i).^2)).*log(x(i+1).^2);
 v(i)=v(i)+4*x(i).*((x(i+1).^2).^(x(i).^2)).*x(i+1);
 v1=v;
 i=[(1:n)';(1:(n-1))'];
 j=[(1:n)';(2:n)'];
 s=[v0;2*v1];
H=sparse(i,j,s,n,n);
 H=(H+H')/2;
End

%调用求非线性最小函数 fminunc
n = 1000;
xstart = -ones(n,1);
xstart(2:2:n,1) = 1;
options = optimset('GradObj','on','Hessian','on');
[x,fval,exitflag,output] = fminunc(@brownfgh,xstart,options);
```

**【说明】**

这个含有 1 000 个变量的问题经过 8 步算法迭代和 7 步共轭梯度迭代得到结果，最后的目标函数值和收敛速度都接近于 0。对于 fminunc，一阶最优性是函数的梯度无穷大标准，函数的梯度值在局部最优时为 0。

**【结果输出】**

```
exitflag =
            1
fval =
            2.8709e-017
output.iterations
ans =
            8
output.cgiterations
ans =
            7
output.firstorderopt
ans =
            4.7948e-010
```

### 2. 用有限差分规则提供海色矩阵稀疏结构求解

我们还可以用有限差分规则提供海色矩阵稀疏结构来求解此类非线性最小化问题。以下我们解决同样的问题，海色矩阵被近似的稀疏有限差分矩阵代替。为在 fminunc 中采用大规模算法，必须计算函数的梯度（在中小规模系统算法中这是不需要的）。

**【程序清单】**

例程 4-5 是求解的 MATLAB 代码。

例程 4-5

```
%第一步：写一个函数文件 brownfg.m 计算目标函数和函数梯度
function [f,g] = brownfg(x,dummy)
% 计算目标函数值
n=length(x); y=zeros(n,1);
i=1:(n-1);
y(i)=(x(i).^2).^(x(i+1).^2+1) + (x(i+1).^2).^(x(i).^2+1);
f=sum(y);

% 计算函数梯度
if nargout > 1
i=1:(n-1); g = zeros(n,1);
g(i) = 2*(x(i+1).^2+1).*x(i).*((x(i).^2).^(x(i+1).^2))+
      2*x(i).*((x(i+1).^2).^(x(i).^2+1)).*log(x(i+1).^2);
g(i+1) = g(i+1) + 2*x(i+1).*((x(i).^2).^(x(i+1).^2+1)).* ...
log(x(i).^2) +2*(x(i).^2+1).*x(i+1).* ((x(i+1).^2).^ (x(i) ^2));
end

%第二步：调用求非线性最小函数 fminunc
fun = @brownfg;
% Get Hstr, structure of the Hessian
load brownhstr
% View the sparsity structure of Hs
spy(Hstr) tr
n = 1000;
xstart = -ones(n,1);
xstart(2:2:n,1) = 1;
options = optimset('GradObj','on','HessPattern',Hstr);
[x,fval,exitflag,output] = fminunc(fun,xstart,options);
```

**【说明】**

为了使稀疏差分矩阵进行有效的计算，必须首先提供海色矩阵的稀疏结构。在这个例子中，假设稀疏矩阵 ***Hstr*** 存在文件 brownhstr.mat 中。使用 optimset 命令设置海色稀疏模型参数。当一个问题有明显的稀疏结构时，如果不设置海色稀疏模型参数，那么在计算时 fminunc 将采用稠密有限差分矩阵形式，这将浪费大量的计算机内存，甚至会溢出。因为此命令使用了函数的梯度后，必须设置 GradObj 参数为'on'。现在执行此 fminunc 命令得

到结果。

**【结果输出】**

```
exitflag =
        1
fval =
        7.4738e-017
output.iterations
ans =
        8
output.cgiterations
ans =
        7
output.firstorderopt
ans =
        7.9822e-010
```

### 3．对具有端约束的非线性优化问题的求解

例如，对此问题：

$$\min \quad f(\boldsymbol{x}) = 1 + \sum_{i=1}^{n} \left|(3-2x_i)x_i - x_{i-1} - x_{i+1} + 1\right|^p + \sum_{i=1}^{n/2} \left|x_i + x_{i+n/2}\right|^p$$

$$n = 800,\ \ p = 7/3$$

$$s.t. -10.0 \leqslant x_i \leqslant 10.0,\ \ x_0 = x_n = 0$$

**【程序清单】**

例程 4-6 是求解的 MATLAB 代码

例程 4-6

```
%第一步：写一个文件 tbroyfg.m，它计算目标函数和它的梯度
n=length(x);   %n 必是 4 的幂
p=7/3; y=zeros(n,1);
i=2:(n-1);
y(i)= abs((3-2*x(i)).*x(i)-x(i-1)-x(i+1)+1).^p;
y(n)= abs((3-2*x(n)).*x(n)-x(n-1)+1).^p;
y(1)= abs((3-2*x(1)).*x(1)-x(2)+1).^p;
j=1:(n/2); z=zeros(length(j),1);
z(j)=abs(x(j)+x(j+n/2)).^p;
f=1+sum(y)+sum(z);
% 计算梯度
if nargout > 1
    p=7/3; n=length(x); g = zeros(n,1); t = zeros(n,1);
    i=2:(n-1);
    t(i)=(3-2*x(i)).*x(i)-x(i-1)-x(i+1)+1;
```

```
        g(i)= p*abs(t(i)).^(p-1).*sign(t(i)).*(3-4*x(i));
        g(i-1)=g(i-1)-p*abs(t(i)).^(p-1).*sign(t(i));
        g(i+1)=g(i+1)-p*abs(t(i)).^(p-1).*sign(t(i));
        tt = (3-2*x(n)).*x(n)-x(n-1)+1;
        g(n)=g(n)+p*abs(tt).^(p-1).*sign(tt).*(3-4*x(n));
        g(n-1)=g(n-1)-p*abs(tt).^(p-1).*sign(tt);
        tt=(3-2*x(1)).*x(1)-x(2)+1;
        g(1)=g(1)+p*abs(tt).^(p-1).*sign(tt).*(3-4*x(1));
        g(2)=g(2)-p*abs(tt).^(p-1).*sign(tt);
        j=1:(n/2); t(j)=x(j)+x(j+n/2);
        g(j) = g(j)+p*abs(t(j)).^(p-1).*sign(t(j));
        jj=j+(n/2);
        g(jj) = g(jj)+p*abs(t(j)).^(p-1).*sign(t(j));
        grad = g;
    end

    %第二步：使用命令 fmincon 来解决此问题
    fun = @tbroyfg;
    load tbroyhstr
    %得到稀疏矩阵结构
    n = 800;
    xstart = -ones(n,1); xstart(2:2:n) = 1;
    lb = -10*ones(n,1); ub = -lb;
    options = optimset('GradObj','on','HessPattern',Hstr);
    [x,fval,exitflag,output] = ...
    fmincon(fun,xstart,[],[],[],[],lb,ub,[],options);
```

**【结果输出】**

```
exitflag =
        1
fval =
        270.4790
output =
iterations:     8
funcCount:      8
cgiterations:   18
firstorderopt: 0.0163
algorithm: 'large-scale: trust-region reflective Newton'
```

Hessian 矩阵稀疏模型预存到文件 tbroyhstr.mat 中。但是此稀疏矩阵结构是有限的，如图 4-1 所示。

在 MATLAB 命令主窗口中输入：

```
load tbroyhstr
spy(Hstr)
```

我们可以观察到它的稀疏结构。

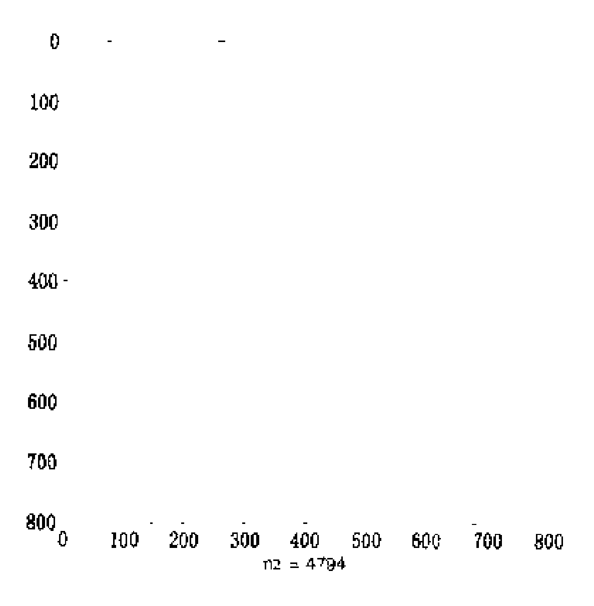



图 4-1　Hessian 矩阵稀疏模型

中间的一条对角线实际上是由 5 条线组成的，使用下面的命令可以看得更清楚，如图 4-2 所示：

```
spy(Hstr(1:20,1:20))
```

使用 optimset 命令设置海色稀疏模型参数、当一个问题有明显的稀疏结构时，如果不设置海色稀疏模型参数，那么在计算时 fminunc 将采用稠密有限差分矩阵形式，这将不必要地浪费大量的计算机内存，甚至会溢出

因为此命令也使用了函数的梯度，所以必须设置 GradObj 参数为'on'。

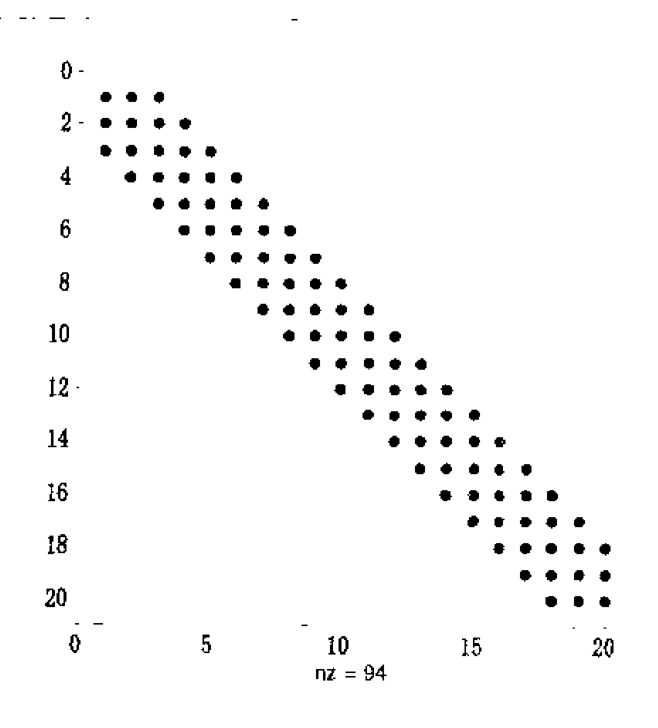



图 4-2　中间的一条对角线

【说明】

为了使稀疏差分矩阵进行有效的计算，必须预先提供海色矩阵的稀疏结构。在这个例子中，假设稀疏矩阵 ***Hstr*** 存在文件 brownhstr.mat 中。使用 optimset 命令设置海色稀疏模型参数。当一个问题有明显的稀疏结构时，如果不设置海色稀疏模型参数，那么在计算时 fminunc 将采用稠密有限差分矩阵形式，这将不必要地浪费大量的计算机内存，甚至会溢出。

因为此命令也使用了函数的梯度，所以必须设置 GradObj 参数为'on' 。现在执行此 fminunc 命令得到结果。

因为有 5 个带状中心对角线，所以使用 five-banded preconditioner 而不是默认的对角线处理。

使用 optimset 命令设置 PrecondBandWidth 参数为 2，重新来解这个问题（带宽是最上面或最下面的对角线的数目，而不包括中间的对角线）。

```
fun = @tbroyfg;
load tbroyhstr % Get Hstr. structure of the Hessian
n = 800;
xstart = -ones(n,1); xstart(2:2:n,1) = 1;
lb = -10*ones(n,1); ub = -lb;
options = optimset('GradObj','on','HessPattern',Hstr, ...
'PrecondBandWidth',2);
[x,fval,exitflag,output] = ...
fmincon(fun,xstart,[],[],[],[],lb,ub,[],options);
```

【结果输出】

```
exitflag =
          1
fval =
          2:7048e+002
output =
     iterations: 10
     funcCount: 10
     cgiterations: 15
     firstorderopt: 7.5339e-005
     algorithm: 'large-scale: trust-region reflective Newton'
```

显然，迭代次数增加了 2 步，然而共轭梯度的迭代次数却从 18 降到 15。一阶优化性测量值也减少了 1e-3。

4．具有等式约束的非线性最小化问题

fmincon 函数在没有其他的约束条件时，可以处理等式约束问题。假如求解问题：

$$\min f(x)=\sum_{i=1}^{n-1}\left[\left(x_i^2\right)^{\left(x_{i+1}^2+1\right)}+\left(x_{i+1}^2\right)^{\left(x_i^2+1\right)}\right],\qquad n=1000$$

$$s.t\quad Aeq*x=Beq$$

$n$=1000，***Aeq*** 有 100 个等式。



【程序清单】

例程 4-7 是求解的 MATLAB 代码。

例程 4-7

```
%第一步：写一个.m 文件——brownfgh.m，计算目标函数和目标函数的梯度，以及稀疏海色矩阵
function [f,g,H] = brownfgH(x)
%计算目标函数
  n=length(x); y=zeros(n,1);
  i=1:(n-1);
  y(i)=(x(i).^2).^(x(i+1).^2+1)+(x(i+1).^2).^(x(i).^2+1);
  f=sum(y);
%计算函数梯度矩阵
  if nargout > 1
     i=1:(n-1); g = zeros(n,1);
     g(i)= 2*(x(i+1).^2+1).*x(i).*((x(i).^2).^(x(i+1).^2))+...
             2*x(i).*((x(i+1).^2).^(x(i).^2+1)).*log(x(i+1).^2);
     g(i+1)=g(i+1)+2*x(i+1).*((x(i).^2).^(x(i+1).^2+1)).* ...
log(x(i).^2)+2*(x(i).^2+1).*x(i+1).*((x(i+1).^2).^(x(i).^2))
  end
%
% 计算稀疏对称海色矩阵
if nargout > 2
v=zeros(n,1);
for i=1.(n-1);
v(i)=2*(x(i+1).^2+1).*((x(i).^2).^(x(i+1).^2))+4*(x(i+1).^2+1).*(x(i+1).^2).*(x(i).^2).*((x(i).^2).^ ..
((x(i+1).^2)-1))+ 2*((x(i+1).^2).^(x(i).^2+1)).*(log(x(i+1).^2));
v(i)=v(i)+4*(x(i).^2).*((x(i+1).^2).^(x(i).^2+1)).*((log(x(i+1).^2)).^2);
v(i+1)=v(i+1)+2*(x(i).^2).^(x(i+1).^2+1).*(log(x(i).^2))+.4*(x(i+1).^2).*((x(i).^2).^(x(i+1).^2+1)).*

.((log(x(i).^2)).^2)+2*(x(i).^2+1).*((x(i+1).^2).^(x(i).^2));
v(i+1)=v(i+1)+4*(x(i).^2+1).*(x(i+1).^2).*(x(i).^2).*((x(i+1).^2).^(x(i).^2-1));
v0=v;
v=zeros(n-1,1);
v(i)=4*x(i+1).*x(i).*((x(i).^2).^(x(i+1).^2))+4*x(i+1).*(x(i+1).^2+1).*x(i).*((x(i).^2).^ ...
(x(i+1).^2)).*log(x(i).^2);
 v(i)=v(i)+4*x(i+1).*x(i).*((x(i+1).^2).^(x(i).^2)).*log(x(i+1).^2);
v(i)=v(i)+4*x(i).*((x(i+1).^2).^(x(i).^2)).*x(i+1);
v1=v;
i=[(1:n)';(1:(n-1))'];
     j=[(1:n)';(2:n)'];
s=[v0;2*v1];
H=sparse(i,j,s,n,n);
H=(H+H')/2;
end
end
%第二步：使用 fmincon 命令解决此问题
```

```
fun = @brownfgh;
load browneq % 得到 A 和 B 的等式
n = 1000;
xstart = -ones(n,1); xstart(2:2:n) = 1;
options = optimset('GradObj','on','Hessian','on', ...
'PrecondBandWidth', inf);
[x,fval,exitflag,output] = ...
fmincon(fun,xstart,[],[],Aeq,beq,[],[],[],options);
```

【说明】

令PrecondBandWidth为inf，使用直接解法。

变量*exitflag*表示在16步的迭代后，算法收敛于最后的函数值。

因为我们使用brownfgh来计算目标函数值、梯度和海色矩阵，所以在使用optimset 命令时，需要使用GradObj 和海色矩阵参数。我们把稀疏矩阵***Aeq***和向量***beq***存于文件browneq.mat中，使用load browneq命令可以装入它们的值。

线性约束系统是一个100～1000维、未构造的稀疏结构（可用spy(Aeq)来观察此稀疏结构），但并不是病态的。

```
condest(Aeq*Aeq')
ans =
    2.9310e+006
```

【结果输出】

```
exitflag =
        1
fval =
        204.9313
output =
iterations: 16
funcCount: 16
cgiterations: 14
firstorderopt: 2.1434e-004
algorithm: 'large-scale: projected trust-region Newton'
%带入求得结果，线性等式接近为 0，说明结果是令人满意的
norm(Aeq*x-beq)
ans =
        1913e-012
```

5．具有稠密结构海色矩阵的非线性最优化问题

大规模优化算法函数fmincon和fminunc可以解决海色矩阵为稠密但可构造的问题。对于这些问题，fmincon和fminunc并不像解决中规模问题那样直接计算***H**Y***，因为在大规模问题中，计算机的内存有可能不能承受计算***H***。我们可以为fmincon和fminunc提供一个函数，此函数给出矩阵***Y***和关于海色矩阵的信息，以及计算***W*=*H**Y***，我们来看一个例子。

在例子中，目标函数是非线性的，约束是线性的。

目标函数为：



$$f(x)=\hat{f}(x)-\frac{1}{2}x^{T}VV^{T}x$$

其中 $V$ 是1000×2 矩阵。函数 $f$ 的海色矩阵是稠密的，但是 $\hat{f}$ 的海色矩阵是稀疏的，令 $\hat{\boldsymbol{H}}$ 为 $\hat{f}$ 的海色矩阵，$\boldsymbol{H}$ 为函数 $f$ 的海色矩阵，则：

$$\boldsymbol{H}=\hat{\boldsymbol{H}}-VV^{T}$$

【分析】

为在直接计算 $\boldsymbol{H}$ 时避免内存溢出，此例提供一个海色矩阵乘法函数 hmfleq1。这个函数使用对应于矩阵 $\boldsymbol{Y}$ 的稀疏矩阵 ***Hinfo*** 和 $V$ 去计算海色矩阵乘积 $\boldsymbol{W}$。

$$\boldsymbol{W}=\boldsymbol{H}*\boldsymbol{Y}=(\boldsymbol{Hinfo}-VV^{T})*\boldsymbol{Y}$$

从上式可知，此函数需要 $\hat{\boldsymbol{H}}$ 和 $V$ 来计算海色矩阵的乘积。$V$ 是一个常量，故可以作为一个参数提供给函数 fmincon，然后函数 fmincon 可把此参数传递给函数 hmfleq1。

但 $\hat{\boldsymbol{H}}$ 不是一个常量，故它必须在 $\boldsymbol{x}$ 变化时计算其值。它既可以在计算目标函数时计算，然后作为输出参数输出；也可以在 optimset 中设置'Hessian' 参数为 on'，让函数 fmincon 从目标函数中得到 ***Hinfo*** 的值传递给海色矩阵做乘积，计算函数 hmfleq1。

【程序清单】

例程 4-8 是求解的 MATLAB 代码.

例程 4-8

```
%第一步：写一个.m 文件——brownvv.m，计算目标函数值、梯度和海色矩阵值
%brownvv.m 是计算目标函数文件，它需传递给优化函数 fmincon
%type brownvv
%因为我们使用 brownfgh 计算目标函数值，梯度和海色矩阵，你需要在使用 optimset 命
%令时使用 GradObj 和海色矩阵参数

%第二步：定义一个函数 hmfleq1，它利用在 ibrownvv 中计算的矩阵 Hinfo 和 V，计算海
%色矩阵乘积 W
W = H*Y = (Hinfo - V*V')*Y.
%此函数的形式为：
W = hmfleq1(Hinfo,Y,p1,p2...)
%第一个参数即为函数返回的第三个参数，第二个参数为矩阵 Y
%因为函数 fmincon 使用参数 Y 去计算海色矩阵的乘积，Y 是一个 n 行矩阵，n 即为
%此问题的维数。Y 的列数是可变的。最后，任何附加的参数都会传递给函数 hmfleq1，
%于是 hmfleq1 接受了相同的参数，例如矩阵 V
```

```
function W = hmfleq1(Hinfo,Y,V);
W = Hinfo*Y - V*(V'*Y);

%第三步：使用命令 fmincon 解决此问题。
%从文件中取得 V, Aeq, beq 的值。
%使用命令 optimset 设置 GradObj 和 Hessian 参数为 'on' ，设定 HessMult
%参数为函数句柄，指向函数 hmfleq1。令目标函数和矩阵 V 为优化函数的参数，计算最优结
%果。
% 得到 V, Aeq, beq 值
load fleq1
% 问题维数
n = 1000;
mtxmpy = @hmfleq1; % Function handle to function hmfleq1
xstart = -ones(n,1);
xstart(2:2:n,1) = ones(length(2:2:n),1);
options = optimset('GradObj','on','Hessian','on',...
'HessMult',mtxmpy,'Display','iter');
[x,fval,exitflag,output] = fmincon(@brownvv,xstart,[],[],Aeq,beq,[],[],[], ... options,V);
```

既然我们在 optimset 中令每次迭代的结果都要显示，故在主窗口中，我们可看到结果输出如下。

【结果输出】

| Iteration | f(x) | Norm of step | First-order optimality | CG-iterations |
|---|---|---|---|---|
| 1 | 1997.07 | 1 | 555 | 0 |
| 2 | 1072.56 | 6.31716 | 377 | 1 |
| 3 | 480.232 | 8.19554 | 159 | 2 |
| 4 | 136.861 | 10.3015 | 59.5 | 2 |
| 5 | 44.3708 | 9.04697 | 16.3 | 2 |
| 6 | 44.3708 | 100 | 16.3 | 2 |
| 7 | 44.3708 | 25 | 16.3 | 0 |
| 8 | -8.90967 | 6.25 | 28.5 | 0 |
| 9 | -318.486 | 12.5 | 107 | 1 |
| 10 | -318.486 | 12.5 | 107 | 1 |
| 11 | -414.445 | 3.125 | 73.9 | 0 |
| 12 | -561.688 | 3.125 | 47.4 | 2 |
| 13 | -784.326 | 6.25 | 126 | 3 |
| 14 | -784.326 | 4.30584 | 126 | 5 |
| 15 | -804.414 | 1.07646 | 26.9 | 0 |
| 16 | -822.399 | 2.16965 | 2.8 | 3 |
| 17 | -823.173 | 0.40754 | 1.34 | 3 |
| 18 | -823.241 | 0.154885 | 0.555 | 3 |
| 19 | -823.246 | 0.0518407 | 0.214 | 5 |
| 20 | -823.246 | 0.00977601 | 0.00724 | 6 |

【说明】

我们成功地得到了优化结果。在优化进程中，PCG 迭代收敛是非常快的。经检查，结果是可信的。

```
norm(Aeq*x-beq) =1.2861e-013
```

函数 fmincon 不能使用矩阵 ***H*** 计算先决矩阵，因为 ***H*** 是隐式矩阵。fmincon 使用 ***Hinfo*** 来代替 ***H*** 计算先决矩阵。因为 ***Hinfo*** 同 ***H*** 有同样的大小且在一定程度上是相似的。如果 ***Hinfo*** 与 ***H*** 不一样大小，fmincon 就不能基于对角大规模矩阵算法计算先决矩阵，即按本例算法将不会得到好的结果。

# 4.5 二次规划问题

本节我们研究大规模系统优化的二次规划问题

## 4.5.1 二次规划问题简介

在二次规划中，$q(x)$ 为二次等式。

$$q(\boldsymbol{x})=\frac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x}+f^T\boldsymbol{x} \quad \text{（式 4.9）}$$

子空间信赖域法用来决定搜索方向。然而，在非线性最优化问题中，常常用反射步长代替限制步长，在每一步迭代中，采用反射方向搜索

## 4.5.2 范例分析

### 1．具有端约束的二次规划问题

当求解有上下界约束条件的二次方程的极值时，可以使用MATLAB中的函数quadprog 来解决。例如，求解一个正有限二次问题，其中：函数海色矩阵是对角矩阵，有上下约束条件.

装入海色矩阵，定义 ***f, lb, ub***，可以用命令 quadprog 解决此问题。

【程序清单】

例程 4-9 是求解的 MATLAB 代码.

例程 4-9

```
% 得到H
load qpbox1
lb = zeros(400,1);
lb(400) = -inf;
ub = 0.9*ones(400,1);
ub(400) = inf;
```

```
f = zeros(400,1);
f([1 400]) = -2;
xstart = 0.5*ones(400,1);
[x,fval,exitflag,output]=quadprog(H,f,[],[],[],[],lb,ub,xstart);
```

【结果输出】

```
exitflag =
          1
output =
     firstorderopt: 7.8435e-006
     iterations: 20
     cgiterations: 1809
     algorithm: 'large-scale: reflective trust-region'
```

【说明】

由结果可以看到在收敛前有20次迭代，CG迭代数的最高值表明此函数解决线性系统的计算量非常大。为了减小计算量，一个策略是限制在每次迭代过程中CG迭代的次数。默认数为解决问题的维数的一半，对此问题为200，通过设置MaxPCGIter标志，我们限制CG迭代次数为50。

```
options = optimset('MaxPCGIter',50);
[x,fval,exitflag,output] =quadprog(H,f,[],[],[],[],lb,ub,xstart,options);
```

可以看到，此算法是收敛的，并且CG迭代次数已经下降了。

```
exitflag =
          1
output =
     firstorderopt: 2.3821e-005
     iterations: 36
     cgiterations: 1547
     algorithm: 'large-scale: reflective trust-region'
```

第二个策略是在每次迭代中使用直接解法。我们设置 PrecondBandWidth 为 inf。

```
options = optimset('PrecondBandWidth',inf);
[x,fval,exitflag,output] = ...
quadprog(H,f,[],[],[],[],lb,ub,xstart,options);
```

现在迭代次数下降到 10

```
exitflag =
          1
output =
     firstorderopt: 4.8955e-007
     iterations: 10
     cgiterations: 9
     algorithm: 'large-scale: reflective trust-region'
```



*在每一步使用直接算法可以使迭代的数目下降，但每一步的迭代需要花更多的时间，对于此问题，折中是有益的，此算法花费的时间相对降低了 10 倍。*



2．具有稠密结构海色矩阵的二次规划问题

大规模算法函数fmincon 和fminunc可以解决海色矩阵稠密但可构造的问题。对于这些问题，fmincon 和fminunc并不像解决中规模问题那样直接计算$H*Y$，因为在大规模问题中，计算机的内存可能不能承受计算$H$。我们可以为fmincon 和fminunc提供一个函数，此函数给出矩阵$Y$和关于海色矩阵的信息，以及计算$W=H*Y$，我们来看一个例子。

在这个例子中，$H$ 有此结构 $H = B + A*A'$，$B$ 是一个稀疏 512×512 对称矩阵，$A$ 是由一些稠密列组成的一个 521×10 的稀疏矩阵。为在直接计算 $H$ 时避免内存溢出，此例提供一个海色矩阵乘法函数 pbox4mult。当矩阵 $Y$ 传递给这个函数时，它利用稀疏矩阵 $A$ 和 $B$ 去计算海色矩阵乘积 $W$。

$$W = H*Y = (B + A*A')*Y$$

pbox4mult函数需要$A$和$B$计算海色矩阵的乘积，这里有两个办法可以解决：一是把自变量$A$和$B$传给函数quadprog，也可传给海色相乘函数；二是$A$和$B$可以作为附加参数提供给函数fmincon，然后函数fmincon可把此参数传递给函数hmfleq1。

**【程序清单】**

例程 4-10 是求解的 MATLAB 代码。

例程 4-10

```
%第一步：决定哪个变量作为第一参数传递给函数 quadprog。
%A 或 B 作为第一自变量传给 quadprog。此例中，我们把 B 作为第一自变量，于是 A
%作为附加参数传给它。
quadprog(B,f,[],[],[],[],l,u,xstart,options,A)

第二步：写一个计算海色矩阵乘积的函数。
%现在，定义一个使用 A 和 B 计算海色乘积的函数 qpbox4mult
W = qpbox4mult(Hinfo,Y,p1,p2...)
%qpbox4mult 必须接受传给函数 quadprog 相同的第一自变量，即 qpbox4mult
%需选 B 作为第一自变量
%第二个自变量为矩阵 Y，因为函数 f quadprog 使用参数 Y 去计算海色矩阵的乘积，
%Y 是一个 n 行矩阵，n 即为此问题的维数，Y 的列数是可变的。最后，任何附加的
%参数都会传递给函数 qpbox4mult，于是 qpbox4mult 接受了相同的附加参数，例
%如矩阵 A。

function W = qpbox4mult(B,Y,A);
W = B*Y + A*(A'*Y);

第三步：使用二次规划函数 quadprog 解此问题
%从文件 qpbox4.mat 中装入参数，使用 optimset 设置 HessMult 为函数
%qpbox4mult 的局柄，B 作为第一参数，而 A 作为附加参数。

load qpbox4 % Get xstart, u, l, B, A, f
mtxmpy = @qpbox4mult; % Function handle to function qpbox4mult
options = optimset('HessMult',mtxmpy);
```

```
[x,fval,exitflag,output] = quadprog(B,f,[],[],[],[],l,u, xstart,options,A);
```



目标函数的相对变化少于 sqrt(Options.TolFun)，在迭代中，矩阵 hesse 没有负曲率存在，所以目标函数改变非常缓慢，经过 18 步迭代，伴随 30 步 PCG 迭代，目标函数值减为-1.0538e+003。

**【结果输出】**

```
fval =
        -1.0538e+003
output.firstorderopt =
        0.0043
```

**【说明】**

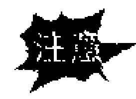
在此例中，quadprog不能使用$H$矩阵计算预处理矩阵，这是因为$H$是隐性存在的；所以quadprog使用矩阵$B$来计算预处理矩阵（$B$具有与$H$相同的大小，并且在一定程度上逼近于$H$）。quadprog可以计算基于一些对角矩阵的预处理矩阵，但一般这并不能总实现得很好。

既然预处理矩阵可以更加逼近$H$，则当$H$为隐性时，可以改变参数TolPcg，使其比原来更小一点，例如减少TolPcg，从0.1变为0.01。计算结果如下：

```
options = optimset('HessMult',mtxmpy,'TolPcg',0.01);
[x,fval,exitflag,output]=quadprog(B,f,[],[],[],[],l,u, xstart,options,A);
```

**【结果输出】**

```
fval =
        -1.0538e+003
output.firstorderopt =
        0.0028
```



减小 TolPcg 必引起 PCG 的迭代次数的增加。

# 4.6　线性最小二乘问题

本节我们研究大规模系统优化的线性最小二乘问题。

## 4.6.1　线性最小二乘问题简介

对（式 4.10）表示的问题可以用最小二乘法解决。

$$\min f(x)=\frac{1}{2}\|Cx+d\|_2^2 \quad \text{（式 4.10）}$$

这个算法在一定的限制条件下可以做到局部严格迭代收敛，每次迭代包括一次大规模线性系统的逼近解。迭代矩阵具有矩阵 $C$ 的结构，此算法采用共轭梯度法来解决非病态的等式

$$C^T Cx = -C^T d$$

此等式不是显式存在的。

使用了空间信赖域法来决定搜索方向，用反射步长代替限制步长，在每一步迭代中，采用反射方向搜索。

### 4.6.2　范例分析

许多问题可以归结为变量具有限制条件的稀疏线性最小二乘问题。我们求解的例子中要求变量是非负的，此问题是：要找到一个函数使之逼近一个分段线性样条，点在单元矩阵中是分散的。函数通过对点的评价来逼近得到，分段线性样条在系数为非负时实时逼近。此问题由400个变量组成的2 000个等式构成，可以用load particle得到问题的 **C**，**D** 。

下面介绍一个具有端约束的线性最小二乘问题的例子

**【程序清单】**

例程 4-11 是求解的 MATLAB 代码。

例程 4-11

```
load particle % Get C, d
lb = zeros(400,1);
[x,resnorm,residual,exitflag,output] = ...
lsqlin(C,d,[],[],[],[],lb);
```

**【结果输出】**

```
exitflag =
        1
resnorm =
        22.5794
output =
        algorithm: 'large-scale: trust-region reflective Newton'
        firstorderopt: 2.7870e-005
        iterations: 10
        cgiterations: 42
```

**【其他解法】**

对于范围限制问题，一阶优化是 $v*g$ 的无限标准，其中 $v$ 定义为“黑箱约束”，$g$ 是梯度。一阶优化通过使用稀疏QR因子分解得到改善，设置PrecondBandWidth为inf。

```
options = optimset('PrecondBandWidth',inf);
[x,resnorm,residual,exitflag,output] =lsqlin(C,d,[],[],[],[],lb,[],[],options);
```

**【结果输出】**

```
exitflag =
        1
resnorm =
```

```
        22.5794
output =
      algorithm: 'large-scale: trust-region reflective Newton'
      firstorderopt: 4.5907e-015
      iterations: 12
      cgiterations: 11
```

结果表明，迭代次数和一阶优化都下降了。

# 4.7 大规模线性优化问题

本节我们研究大规模线性优化问题。

## 4.7.1 大规模线性优化问题简介

对于大规模线性规划问题

$$\min f^T x$$
$$s.t\begin{cases} Aeqx = beq \\ Aineqx = bineq \\ l \leqslant x \leqslant u \end{cases} \quad （式 4.11）$$

MATLAB 使用的求解算法是基于 LIPSOL 的，它是由 Mehrotra 的预报校正算法变化而来的，是一种原始对偶内点法。

它的主要算法原理介绍如下。

首先把问题化为线性规划的标准形式

$$\min \quad f^T x$$
$$s.t. \quad \begin{pmatrix} Ax = b \\ 0 \leqslant x \leqslant u \end{pmatrix} \quad （式 4.12）$$

假如一个松散变量，使其去掉上限。（式 4.12）变为

$$\min \quad f^T x$$
$$s.t. \quad \begin{pmatrix} Ax = b \\ x + s = u \\ x \geqslant 0, s \geqslant 0 \end{pmatrix} \quad （式 4.13）$$

$x$ 为需求解的原始变量，$s$ 为松散变量。它的对偶问题为

$$\max \quad b^T y - u^T w$$
$$s.t. \quad \begin{pmatrix} A^T y - w + z = f \\ z \geqslant 0, w \geqslant 0 \end{pmatrix} \quad （式 4.14）$$



$y$ 和 $w$ 为对偶原始变量，$z$ 为对偶松散变量。

此线性规划的优化条件为：

$$F(x,y,z,s,w)=\begin{pmatrix} Ax-b \\ x+s-u \\ A^T y-w+z-f \\ x_i z_i \\ s_i w_i \end{pmatrix}=0 \qquad （式4.15）$$

$$x\geq 0, y\geq 0, z\geq 0, s\geq 0, w\geq 0$$

二次方程式 $x_i z_i=0$ 和 $s_i w_i=0$ 为线性规划的强制性条件，其他的等式为线性规划的可行性条件，此时 $x^T z+s^T w$ 的大小为对偶间隙，它是用来测量 $F$ 余部残差的大小的量。

此算法可以同时解决原始规划和其对偶规划问题。它对线性二次系统（式 4.12）使用牛顿法，但在迭代过程中保持迭代的 $x,z,w$ 和 $s$ 为正，这也是为什么称为内点法的原因，迭代在（式 4.12）中的约束不等式表示内部区域进行。

此算法是由预报校正算法发展而来的，它是由 Mehrotra 提出的。考虑一次迭代过程 $v=[x;\ y;\ z;\ s;\ w]$，计算其预测搜索方向

$$\nabla v_p=-\left(F^T(v)\right)^{-1}F(v)$$

这是牛顿方向，其校正方向为

$$\nabla v_c=-\left(F^T(v)\right)^{-1}\left(F\left(v+\nabla v_p\right)\right)-\mu\hat{e}$$

其中 $\mu$ 为精心选择的定中心参数，$\hat{e}$ 是对应于二次等式的 0－1 向量，即它只针对于二次强制性条件。使用步长参数 $\alpha>0$ 连接两个搜索方向，则新的迭代方向为：

$$v^+=v+\alpha\left(\Delta v_p+\Delta v_c\right)$$

即 $v^+=\left[x^+;\ y^+;\ z^+;\ s^+;\ w^+\right]$，满足 $\left[x^+;\ z^+;\ s^+;\ w^+\right]>0$

为解决搜索步长，此算法对 $F^T$ 的 Cholesky 因子进行直接稀疏因子分解，如果 $A$ 有稠密的列，它使用 Sherman-Morriso 公式；如果结果是不充分的（残差很大），它使用共轭梯度法。

此算法重复以上步骤直到其收敛，算法的结束准则为：

$$\frac{\|r_b\|}{\max(1,\ \|b\|)}+\frac{\|r_f\|}{\max(1,\ \|f\|)}+\frac{\|r_u\|}{\max(1,\ \|u\|)}+\frac{|f^Tx-b^Ty+u^Tw|}{\max(1,\ |f^Tx|\ \ |b^Ty-u^Tw|)}\leqslant tol$$

其中

$$r_b = Ax - b$$

$$r_f = A^Ty - w + z - f$$

$$r_u = x + s - u$$

分别是原始残差、对偶残差、最大限可行性。

$f^Tx - b^Ty + u^Tw$ 是原始规划和对偶规划目标值之差。$tol$ 是容忍误差，它是所有优化条件的误差之和。

### 4.7.2 范例分析

这一部分我们主要讲述两个大规模线性优化问题，即具有等式和不等式约束的线性规划问题和具有稠密列的等式线性规划问题。

**1. 具有等式和不等式约束的线性规划问题**

此问题是

$$\min f^Tx$$

$$s.t\begin{cases} Aeqx = beq \\ Ax \leqslant b \\ x \geqslant 0 \end{cases}$$

此问题可以在 MATLAB 中用 load sc50b 命令装入向量 $A, Aeq, b, beq, f$ 值和约束 $lb$。

load sc50b

此问题有48个变量，30个不等式约束和20个等式约束。

我们可以使用函数linprog解决此类问题。

**【程序清单】**

例程 4-12 是求解的 MATLAB 代码。

例程 4-12

```
load sc50b
[x,fval,exitflag,output]=linprog(f,A,b,Aeq,beq,lb,[], ...
[],optimset('Display','iter'));
```

**【结果输出】**

Residuals: Primal Dual Duality Total
Infeas Infeas Gap Rel

```
A*x-b A'*y+z-f x'*z Error
--------------------------------------------------------
Iter 0: 1.50e+003 2.19e+001 1.91e+004 1.00e+002
Iter 1: 1.15e+002 2.94e-015 3.62e+003 9.90e-001
Iter 2: 1.16e-012 2.21e-015 4.32e+002 9.48e-001
Iter 3: 3.23e-012 4.16e-015 7.78e+001 6.88e-001
Iter 4: 4.78e-011 7.61e-016 2.38e+001 2.69e-001
Iter 5: 9.31e-011 1.84e-015 4.05e+000 6.89e-002
Iter 6: 2.96e-011 1.62e-016 1.64e-001 2.34e-003
Iter 7: 1.51e-011 2.74e-016 1.09e-005 1.55e-007
Iter 8: 1.51e-012 2.37e-016 1.09e-011 1.51e-013
```

对此类问题，大规模规划算法可以很快地减小残差，使它低于默认的容忍误差 1e-08。参数 exitflag 为正，表示函数 linprog 收敛。我们还可以查看最后的目标函数值及迭代次数。

```
exitflag =
        1
fval =
        -70.0000
output =
     iterations: 8
     cgiterations: 0
     algorithm: 'lipsol'
```

2．具有稠密列的等式线性规划问题

此例为：

$$\min f^T x$$
$$s.t.\begin{cases} Aeqx = beq \\ lb \le x \le ub \end{cases}$$

首先可以在 MATLAB 中用 load densecolumns 命令装入向量 $A, Aeq, b, beq, f$ 值和约束 $ub$.

```
load densecolumns
```

此问题由1 667个变量和627个等式组成。所有的变量都有下限，399个变量有上限，且等式矩阵具有稠密行。这可以从Spy函数画出的图形上看出，如图4-3所示。

```
spy(Aeq)
```

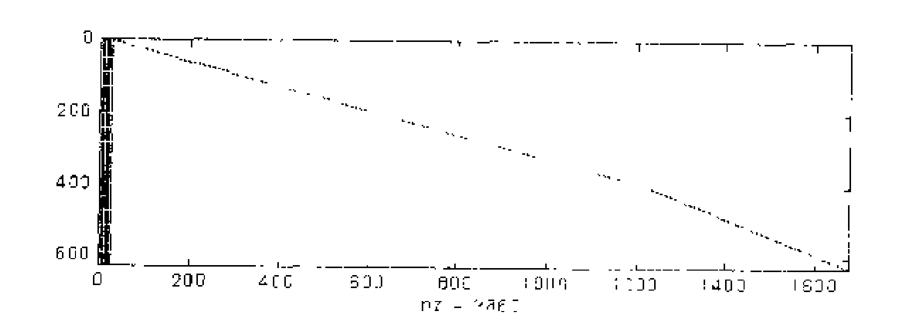



图 4-3　Spy 函数画出的图形

我们可以用函数linprog解决此问题。

【程序清单】

例程 4-13 是求解的 MATLAB 代码。

例程 4-13

```
load densecolumns;
[x,fval,exitflag,output] = ...
linprog(f,[],[],Aeq,beq,lb,ub,[],optimset('Display','iter'));
```

【结果输出】

```
Residuals: Primal Dual Upper Duality Total
Infeas Infeas Bounds Gap Rel
A*x-b A'*y+z-w-f {x}+s-ub x'*z+s'*w Error
-------------------------------------------------------------
Iter 0: 1.67e+003 8.11e+002 1.35e+003 4.30e+006 2.92e+001
Iter 1: 1.37e+002 1.33e+002 1.11e+002 1.27e+006 2.48e+000
Iter 2: 3.56e+001 2.38e+001 2.89e+001 3.42e+005 1.99e+000
Iter 3: 4.86e+000 8.88e+000 3.94e+000 1.40e+005 1.89e+000
Iter 4: 4.24e-001 4.89e-001 3.44e-001 1.91e+004 8.41e-001
Iter 5: 1.23e-001 2.02e-001 9.97e-002 8.41e+003 4.79e-001
Iter 6: 3.98e-002 7.91e-002 3.23e-002 4.05e+003 3.52e-001
Iter 7: 7.25e-003 3.83e-002 4.88e-003 1.85e+003 1.85e-001
Iter 8: 1.47e-003 1.34e-002 1.19e-003 8.12e+002 8.52e-002
Iter 9: 2.52e-004 3.39e-003 2.04e-004 2.78e+002 2.99e-002
Iter 10: 3.46e-005 1.08e-003 2.81e-005 1.09e+002 1.18e-002
Iter 11: 6.95e-007 1.53e-012 4.64e-007 1.48e+001 1.62e-003
Iter 12: 1.04e-006 2.26e-012 3.18e-008 8.32e-001 9.09e-005
Iter 13: 3.08e-006 1.23e-012 3.86e-009 7.26e-002 7.94e-006
Iter 14: 3.75e-007 1.09e-012 6.53e-012 1.11e-003 1.21e-007
Iter 15: 4.21e-008 1.30e-012 3.27e-013 8.62e-008 9.15e-010
```

我们还可以看到最后的目标函数值及迭代次数。

```
exitflag =
        1
fval =
        9.1464e+003
output =
    iterations: 15
    cgiterations: 225
    algorithm: 'lipsol'
```



可以看出 PCG 迭代的次数是非零的，这是因为稠密列被函数检测出。它不再使用 Cholesky 因子，而是使用 Sherman-Morrison 公式去解决此线性问题。如果此公式不能得到满意的残差，它将使用 PCG 迭代法。

# 第 5 章　工程优化算法及其 MATLAB 实现（三）——遗传算法

作为一种新的全局优化搜索算法，遗传算法以其简单通用、鲁棒性强、适于并行处理及高效实用等显著特点，在各个领域得到了广泛应用，取得了良好的效果。本章首先介绍了遗传算法的由来、特点和实现步骤；然后给出了用 MATLAB 实现遗传算法的程序；最后举例说明了遗传算法在工程最优化问题中的应用。

本章主要内容：

- 遗传算法简介
- 遗传算法的 MATLAB 实现
- 遗传算法在一维变量优化中的应用
- 遗传算法在多维变量优化中的应用
- 遗传算法在无约束优化中的应用
- 遗传算法在非线性规划中的应用
- 遗传算法在可靠性优化中的应用
- 遗传算法在车间布局优化中的应用
- 遗传算法在参数优化中的应用
- 遗传算法在动态系统最优控制中的应用

## 5.1　引　　言

在工业工程中，许多最优化问题性质十分复杂，很难用传统的优化方法来求解。自 1960 年以来，人们对求解这类难解问题的兴趣日益增加。一种模仿生物自然进化过程的、被称为“进化算法（evolutionary algorithm）”的随机优化技术在解这类优化难题中显示出了优于传统优化算法的性能。目前，进化算法主要包括三个研究领域：遗传算法、进化规划（evolutionary programming）和进化策略（evolutionary strategies）。其中遗传算法是迄今为止进化算法中应用最多、比较成熟、广为人知的算法。由于其在求解复杂优化问题的巨大潜力及其在工业工程领域的成功应用，这种算法受到了广泛的注意。这些成功的应用包括：作业调度与排序、可靠性设计、车辆路径选择与调度、成组技术、设备布置与分配、交通问题，以及其他的许多优化问题。

# 5.2 遗传算法简介

遗传算法（Genetic Algorithm，缩写为 GA）是一种有效地解决最优化问题的方法。它最先是由 John Holland 于 1975 年提出的。遗传算法是模拟达尔文的遗传选择和自然淘汰的生物进化过程的计算模型。它的思想源于生物遗传学和适者生存的自然规律，是具有“生存＋检测”的迭代过程的搜索算法。遗传算法以一种群体中的所有个体为对象，并利用随机化技术指导对一个被编码的参数空间进行高效搜索。其中，选择、交叉和变异构成了遗传算法的遗传操作；参数编码、初始群体的设定、适应度函数的设计、遗传操作设计、控制参数设定等 5 个要素组成了遗传算法的核心内容。

## 5.2.1 遗传算法的基本步骤

遗传算法是一种基于生物自然选择与遗传机理的随机搜索算法，与传统搜索算法不同，遗传算法从一组随机产生的称为“种群（population）”的初始解开始搜索过程。种群中的每个个体是问题的一个解，称为“染色体（chromosome）”。染色体是一串符号，比如一个二进制字符串。这些染色体在后续迭代中不断进化，称为遗传。在每一代中用“适值（fitness）”来测量染色体的好坏，生成的下一代染色体称为后代（offspring）。后代是由前一代染色体通过交叉（crossover）或者变异（mutation）运算形成的。在新一代形成过程中，根据适度的大小选择部分后代，淘汰部分后代，从而保持种群大小是常数。适值高的染色体被选中的概率较高，这样经过若干代之后，算法收敛于最好的染色体，它很可能就是问题的最优解或次优解。

遗传算法的主要步骤如下所示。

（1）编码：GA 在进行搜索之前先将解空间的解数据表示成遗传空间的基因型串结构数据，这些串结构数据的不同组合便构成了不同的点

（2）初始群体的生成：随机产生 $N$ 个初始串结构数据，每个串结构数据称为一个个体，$N$ 个个体构成了一个群体。GA 以这 $N$ 个串结构数据作为初始点开始迭代。

（3）适应性值评估检测：适应性函数表明个体或解的优劣性。对于不同的问题，适应性函数的定义方式也不同。

（4）选择：选择的目的是为了从当前群体中选出优良的个体，使它们有机会作为父代为下一代繁殖子孙。遗传算法通过选择过程体现这一思想，进行选择的原则是适应性强的个体为下一代贡献一个或多个后代的概率大。选择实现了达尔文的适者生存原则。

（5）交叉：交叉操作是遗传算法中最主要的遗传操作。通过交叉操作可以得到新一代个体，新个体组合了其父辈个体的特性。交叉体现了信息交换的思想。

（6）变异：变异首先在群体中随机选择一个个体，对于选中的个体以一定的概率随机地改变串结构数据中某个串的值。同生物界一样，GA 中变异发生的概率很低，通常取值在 0.001~0.01 之间。变异为新个体的产生提供了机会。

实际上，遗传算法中有两类运算：

- 遗传运算：交叉和变异



● 进化运算：选择

遗传算法模拟了基因在每一代中创造新后代的繁殖过程，进化运算则是种群逐代更新的过程。交叉是最主要的遗传运算，它同时对两个染色体操作，组合两者的特性产生新的后代。交叉的最简单方法是在双亲（两个父辈的个体）的染色体上随机地选择一个断点，将断点的右段互相交换，从而形成两个新的后代。这种方法对于二进制表示最为合适。遗传算法的性能在很大程度上取决于采用的交叉运算的性能。变异则是一种基本运算，它在染色体上自发地产生随机的变化。一种简单的变异方法是替换一个或多个基因。在遗传算法中，变异可以提供初始种群中不含有的基因，或找回选择过程丢失的基因，为种群提供新的内容。

GA 的计算过程流程图如图 5-1 所示。

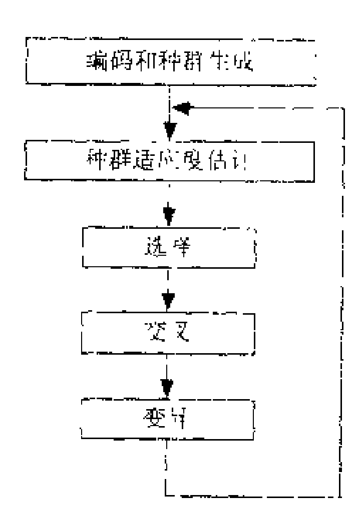



图 5-1　GA 的计算过程流程图

### 5.2.2　遗传算法的特点

GA 算法具有下述特点：

- GA 是对问题参数的编码组进行计算，而不是直接针对参数本身。
- GA 的搜索是从问题解的编码组开始搜索，而不是从单个解开始。
- GA 使用目标函数值（适应度）这一信息进行搜索，而不需导数等其他信息。
- GA 算法使用的选择、交叉、变异这三个算子都是随机操作，而不是确定规则

### 5.2.3　遗传算法在工程优化中的应用

最优化问题通常可归结为极大化问题，利用数学公式描述可以写为：

$$\max_{x\in S} f(x)$$

其中 $f(x)$为目标函数，$S$ 为可行域，它们是由工程实际问题的具体条件决定的。

实践表明，遗传算法解最优化问题的计算效率比较高，适用范围相当广。为了解释这

一现象，Holland 给出了图式定理。所谓图式，就是某些码位取相同值的编码的集合。图式定理说明在进化过程的各代中，属于适应度高、阶数低且长度短的图式的编码数量将随代数以指数形式增长。另外，Holland 还发现遗传算法具有隐含的并行计算特性。最近的研究则表明，上述遗传算法经适当改进后对任意优化问题以概率 1 收敛于全局最优解。

利用遗传算法解最优化问题，首先应对可行域中的点进行编码（一般采用二进制编码），然后在可行域中随机挑选一些编码组作为进化起点的第一代编码组，并计算每个解的目标函数值，也就是编码的适应度。接着就像自然界中一样，利用选择机制从编码组中随机挑选编码作为繁殖过程前的编码样本。选择机制应保证适应度较高的解能够保留较多的样本；而适应度较低的解则保留较少的样本，甚至被淘汰。在接下去的繁殖过程中，遗传算法提供了交叉和变异两种算子对挑选后的样本进行交换。交叉算子交换随机挑选的两个编码的某些位，变异算子则直接对一个编码中的随机挑选的某一位进行反转。这样通过选择和繁殖就产生了下一代编码组。重复上述选择和繁殖过程，直到结束条件得到满足为止。进化过程最后一代中的最优解就是用遗传算法解最优化问题所得到的最终结果。

# 5.3 遗传算法的 MATLAB 实现

前面对遗传算法及其功能进行了简介，那么怎样用 MATLAB 来实现呢？本节将对具体的实现方法进行详细阐述，并穿插了很多的例程。

**1. 编码和种群生成**

通过参数 options 来选择编码的形式：浮点编码或二进制编码。

**【参数说明】**

pop：生成的初始种群

populatoinSize：种群中的个体的数目

variableBounds：表示变量边界的矩阵

eevalFN：适应度函数

eevalOps：传递给适应度函数的参数。

例程 5-1 是求解的 MATLAB 代码。

例程 5-1

```
function [pop] = initializega(num, bounds, eevalFN,eevalOps,options)
if nargin<5
  options=[1e-6 1];
end
if nargin<4
  eevalOps=[];
end
if options(2)==1   %浮点编码
    estr=['[ pop(i,:) pop(i,xZomeLength)]=' eevalFN '(pop(i,:),[0 eevalOps]);'];
  else   %二进制编码
    estr=['x=b2f(pop(i,:),bounds,bits);[x v]=' eevalFN(x,[0 eevalOps]);
```

```
pop(i,:)=[f2b(x,bounds,bits) v];
   end
end

numVars = size(bounds,1);        % 变量的数目
rng= (bounds(:,2)-bounds(:,1))';   % 变量边界

if options(2)==1
   xZomeLength = numVars+1;
   pop= zeros(num,xZomeLength);
   pop(:,1:numVars)=(ones(num,1)*rng).*(rand(num,numVars))+ (ones(num,1)*bounds(:,1)');
else
   bits=calcbits(bounds,options(1));
   xZomeLength = sum(bits)+1;
   pop = round(rand(num,sum(bits)+1));
end

for i=1:num
   eeval(estr);
end
```

2. 交叉

交叉过程选择两个个体作为父代，产生两个新的子代个体。

【参数说明】

p1：第一个父代个体

p2：第二个父代个体

bounds：可行域的边界矩阵。

例程 5-2 是求解的 MATLAB 代码。

例程 5-2

```
function [c1,c2]=crossover(p1,p2,bounds,ops)
sz=size(p1,2)-1;
right=[2:sz 1];
left =[sz 1:(sz-1)];
p1i(p1)=1:sz; %生成序列号
p2i(p2)=1:sz; %生成序列号
adj=sort([-1:-1:-sz;p1(right(p1i));p1(left(p1i));p2(right(p2i));p2(left(p2i))]');
repeats=find(diff(adj(:,2:5)')'==0);
adj(repeats+sz)=zeros(size(repeats));

curr_site = round(rand*sz + 0.5);%选择任意交叉点
for site=1:(sz-1)
    c1(site)=curr_site;
    inAdj=find(adj(:,2:5)==curr_site);
    adj(inAdj+sz)=zeros(size(inAdj));
```

```
        lz=colperm(adj');
        lzi(lz)=1:size(lz,2); %create index
        adj_cities=adj(curr_site,1+(find(adj(curr_site,2:5))));
        [v i]=min(lzi(adj_cities));
        curr_site=adj_cities(i);
    end
    c1(sz)=curr_site;
    c2=p1;
```

3. 变异

变异操作由一个父代产生单个子代。

例程 5-3 是求解的 MATLAB 代码。

例程 5-3

```
function [child] = Mutation(par,bounds,genInfo,Ops)
sz = size(par,2)-1;
ppos = round(rand*sz + 0.5);
cpos = round(rand*sz + 0.5);
if ppos ~= cpos
    if ppos > cpos
        child = [par(1:cpos-1) par(ppos) par(cpos:ppos-1) par(ppos+1:sz) par(sz+1)];
    else
        child = [par(1:ppos-1) par(ppos+1:cpos) par(ppos) par(cpos+1:sz) par(sz+1)];
    end
else
    child = par;
end
```

4. 选择

选择操作决定哪些个体可以进入下一代，程序中采用标准几何分布进行选择。

**【参数说明】**

newPop：由父代种群选出新的种群

oldPop：当前的种群（父代种群）

options：概率。

例程 5-4 是求解的 MATLAB 代码。

例程 5-4

```
function[newPop] = Select(oldPop,options)
q=options(2);                          % 选择最优的概率
e = size(oldPop,2);
n = size(oldPop,1);                    % 种群中个体的数目
newPop = zeros(n,e);                   % 初始化
fit = zeros(n,1);                      % 初始化
x=zeros(n,2);
```

```
x(:,1) =[n:-1:1]';
 [y x(:,2)] = sort(oldPop(:,e));
r = q/(1-(1-q)^n);                    % 标准分布
fit(x(:,2))=r*(1-q).^(x(:,1)-1);      % 生成选择概率
fit = cumsum(fit);                    % 计算选择概率之和
rNums=sort(rand(n,1));
fitIn=1; newIn=1;
while newIn<=n                        % 得到新的个体
  if(rNums(newIn)<fit(fitIn))
    newPop(newIn,:) = oldPop(fitIn,:); % 选择合适的个体
    newIn = newIn+1;
  else
    fitIn = fitIn + 1;
  end
end
```

例程 5-5 是完整的遗传算法函数程序。

例程 5-5

```
function [x,endPop,bPop,traceInfo] = ga(bounds,eevalFN,eevalOps,startPop,opts,...
termFN,termOps,selectFN,selectOps,xOverFNs,xOverOps,mutFNs,mutOps)

n=nargin;
if n<2 | n==6 | n==10 | n==12
  disp('Insufficient arguements')
end
if n<3 %默认进化选择
  eevalOps=[];
end
if n<5
  opts = [1e-6 1 0];
end
if isempty(opts)
  opts = [1e-6 1 0];
end

if any(eevalFN<48) %不用 a .m 文件
  if opts(2)==1 %浮点编码
    e1str=['x=c1; c1(xZomeLength)=', eevalFN ';'];
    e2str=['x=c2; c2(xZomeLength)=', eevalFN ';'];
  else %二进制编码
    e1str=['x=b2f(endPop(j,:),bounds,bits); endPop(j,xZomeLength)=',...
     eevalFN ';'];
  end
else %使用 a .m 文件
  if opts(2)==1 %浮点编码
```

```
        e1str=['[c1 c1(xZomeLength)]=' eevalFN '(c1,[gen eevalOps]);'];
        e2str=['[c2 c2(xZomeLength)]=' eevalFN '(c2,[gen eevalOps]);'];
      else %二进制编码
        e1str=['x=b2f(endPop(j,:),bounds,bits);[x v]=' eevalFN ...
         '(x,[gen eevalOps]); endPop(j,:)=[f2b(x,bounds,bits) v];'];
      end
    end


    if n<6 %默认终止信息
      termOps=[100];
      termFN='maxGenTerm';
    end
    if n<12 %默认的变异信息
      if opts(2)==1 %浮点编码
      mutFNs=['boundaryMutation multiNonUnifMutation nonUnifMutation unifMutation'];
        mutOps=[4 0 0;6 termOps(1) 3;4 termOps(1) 3;4 0 0];
      else %二进制编码
        mutFNs=['binaryMutation'];
        mutOps=[0.05];
      end
    end
    if n<10     %默认的交叉信息
      if opts(2)==1     %浮点编码
        xOverFNs=['arithXover heuristicXover simpleXover'];
        xOverOps=[2 0;2 3;2 0];
      else %二进制编码
        xOverFNs=['simpleXover'];
        xOverOps=[0.6];
      end
    end
    if n<9 %Default select opts only i.e. roullete wheel.
      selectOps=[];
    end
    if n<8     %默认选择信息
      selectFN=['normGeomSelect'];
      selectOps=[0.08];
    end
    if n<6     %默认的算法终止准则
      termOps=[100];
      termFN='maxGenTerm';
    end
    if n<4     %初始种群为空
      startPop=[];
    end
    if isempty(startPop)     %随机生成初始种群
```

```
  startPop=initializega(80,bounds,eevalFN,eevalOps,opts(1:2));
end

if opts(2)==0      % 二进制编码
  bits=calcbits(bounds,opts(1));
end

xOverFNs=parse(xOverFNs);
mutFNs=parse(mutFNs);

xZomeLength  = size(startPop,2);       %Length of the xzome=numVars+fittness
numVar       = xZomeLength-1;      %变量数目
popSize      = size(startPop,1);              %种群中个体数目
endPop       = zeros(popSize,xZomeLength);     %次种群矩阵
c1           = zeros(1,xZomeLength);           % 个体
c2           = zeros(1,xZomeLength);           % 个体
numXOvers    = size(xOverFNs,1);               % 交叉操作次数
numMuts      = size(mutFNs,1);                 %变异操作次数
epsilon      = opts(1);                        %适应度门限值
oeval        = max(startPop(:,xZomeLength)); %初始种群中的最优值
bFoundIn     = 1;
done         = 0;
gen          = 1;
collectTrace = (nargout>3);
floatGA      = opts(2)==1;
display      = opts(3);

while(~done)
  [beval,bindx] = max(startPop(:,xZomeLength));       %当前种群的最优值
  best =   startPop(bindx,:);

  if collectTrace
    traceInfo(gen,1)=gen;                      %当前代
    traceInfo(gen,2)=startPop(bindx,xZomeLength);       %最优适应度
    traceInfo(gen,3)=mean(startPop(:,xZomeLength));    %平均适应度
    traceInfo(gen,4)=std(startPop(:,xZomeLength));
  end

  if ( (abs(beval - oeval)>epsilon) | (gen==1))
    if display
      fprintf(1,'\n%d %f\n',gen,beval);
    end
    if floatGA
      bPop(bFoundIn,:)=[gen startPop(bindx,:)];
    else
      bPop(bFoundIn,:)=[gen b2f(startPop(bindx,1:numVar),bounds,bits)...
```

```
        startPop(bindx,xZomeLength)];
    end
    bFoundIn=bFoundIn+1;
    oeval=beval;
else
    if display
        fprintf(1,'%d ',gen);
    end
end

endPop = feeval(selectFN,startPop,[gen selectOps]);     %选择操作

if floatGA
    for i=1:numXOvers,
        for j=1:xOverOps(i,1),
    a = round(rand*(popSize-1)+1);      %一个父代个体
    b = round(rand*(popSize-1)+1);      %另一个父代个体
    xN=deblank(xOverFNs(i,:));    %交叉函数
    [c1 c2] = feeval(xN,endPop(a,:),endPop(b,:),bounds,[gen... xOverOps(i,:)]);

    if c1(1:numVar)==endPop(a,(1:numVar))
        c1(xZomeLength)=endPop(a,xZomeLength);
    elseif c1(1:numVar)==endPop(b,(1:numVar))
        c1(xZomeLength)=endPop(b,xZomeLength);
    else
        eeval(e1str);
    end
    if c2(1:numVar)==endPop(a,(1:numVar))
        c2(xZomeLength)=endPop(a,xZomeLength);
    elseif c2(1:numVar)==endPop(b,(1:numVar))
        c2(xZomeLength)=endPop(b,xZomeLength);
    else
        eeval(e2str);
    end

    endPop(a,:)=c1;
    endPop(b,:)=c2;
        end
    end

    for i=1:numMuts,
        for j=1:mutOps(i,1),
    a = round(rand*(popSize-1)+1);
    c1 = feeval(deblank(mutFNs(i,:)),endPop(a,:),bounds,[gen mutOps(i,:)]);
    if c1(1:numVar)==endPop(a,(1:numVar))
        c1(xZomeLength)=endPop(a,xZomeLength);
```

```
        else
          eeval(e1str);
        end
        endPop(a,:)=c1;
          end
        end

    else      %遗传操作的统计模型
      for i=1:numXOvers,
        xN=deblank(xOverFNs(i,:));
        cp=find(rand(popSize,1)<xOverOps(i,1)==1);
        if rem(size(cp,1),2) cp=cp(1:(size(cp,1)-1)); end
        cp=reshape(cp,size(cp,1)/2,2);
        for j=1:size(cp,1)
       a=cp(j,1); b=cp(j,2);
       [endPop(a,:) endPop(b,:)] = feeval(xN,endPop(a,:),endPop(b,:), bounds,[gen xOverOps(i,:)]);
        end
      end
      for i=1:numMuts
        mN=deblank(mutFNs(i,:));
        for j=1:popSize
          endPop(j,:) = feeval(mN,endPop(j,:),bounds,[gen mutOps(i,:)]);
          eeval(e1str);
        end
      end
    end

    gen=gen+1;
    done=feeval(termFN,[gen termOps],bPop,endPop); %
    startPop=endPop;                %更新种群

    [beval,bindx] = min(startPop(:,xZomeLength));
    startPop(bindx,:) = best;
end

[beval,bindx] = max(startPop(:,xZomeLength));
if display
   fprintf(1,'\n%d %f\n',gen,beval);
end

x=startPop(bindx,:);
if opts(2)==0 %binary
      x=b2f(x,bounds,bits);
bPop(bFoundIn,:)=[gen b2f(startPop(bindx,1:numVar),bounds,bits), startPop(bindx,xZomeLength)];
else
   bPop(bFoundIn,:)=[gen startPop(bindx,:)];
```

```
    end
    if collectTrace
        traceInfo(gen,1)=gen;
        traceInfo(gen,2)=startPop(bindx,xZomeLength); %最优适应度
        traceInfo(gen,3)=mean(startPop(:,xZomeLength)); % 平均适应度
    end
```

【参数说明】

- 输出参数

x：求得最优解

endPop：最终的种群

bPop：最优种群的一个搜索轨迹。

- 输入参数

bounds：代表变量的上下界的矩阵

eevalFN：适应度函数

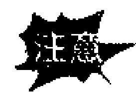
startPop：初始群体

termFN：终止函数的名称

termOps：终止函数的参数

selectFN：选择函数的名称

selectOpts：选择参数。




例程 5-5 的遗传算法求的是函数的极大值，因此在求解极小值问题时需要将极大值问题转换为极小值问题。

# 5.4 范例分析

根据解空间的维数可以将优化问题分为一维空间的优化问题和多维空间的优化问题两大类。本节通过来自这两类问题的两个简单优化问题的求解过程来说明遗传算法在优化中的应用方法和步骤。

## 5.4.1 一维变量优化问题

【问题】

利用遗传算法计算下列函数的最大值

$$f(x) = x + 10*\sin(5x) + 7*\cos(4x)$$
$$x \in [0,9]$$



【分析】

选择二进制编码，种群中的个体数目为 10，二进制编码序列的长度为 20，交叉概率为 0.95，变异概率为 0.08。

【程序清单】

例程 5-6 是求解的 MATLAB 代码

例程 5-6

```
%编写目标函数
function [sol, eval] =gaDemo1Eeval(sol,options)
x=sol(1);
eval = x + 10*sin(5*x)+7*cos(4*x);

%参数说明
%eval：个体的适应度；
%sol：当前个体，n+1 个元素的行向量。

%遗传算法求最大值
clc
fplot('x + 10*sin(5*x)+7*cos(4*x)',[0 9])
% 生成初始种群，大小为 10
initPop=initializega(10,[0 9],'gademo1eeval1').
plot (initPop(:,1),initPop(:,2),'b*')
% 调用遗传函数
% 一次遗传迭代
[x endPop] = ga([0 9],'gademo1eeval1',[],initPop,[1e-5...
1 1],'maxGenTerm',1,'normGeomSelect',[0.08],['arithXover'] ... [20], 'nonUnifMutation',[2 1 3]);

plot (endPop(:,1),endPop(:,2),'bo')
% 25 次遗传迭代
[x endPop bpop trace] = ga([0 9],'gademo1eeval1',[],initPop, [1e-6 1 1],'maxGenTerm',25,
                    'normGeomSelect',[0.08], ['arithXover'],[2],'nonUnifMutation',[2 25 3]);

plot (endPop(:,1),endPop(:,2),'y*')
figure(2)
plot(trace(:,1),trace(:,3),'y-')
hold on
plot(trace(:,1),trace(:,2),'r-')
xlabel('Generation'); ylabel('Fittness');
legend（'解的变化','种群平均值的变化'）；
```

【结果输出】

（1）图 5-2 为目标函数的图形和初始随机种群个体分布图。

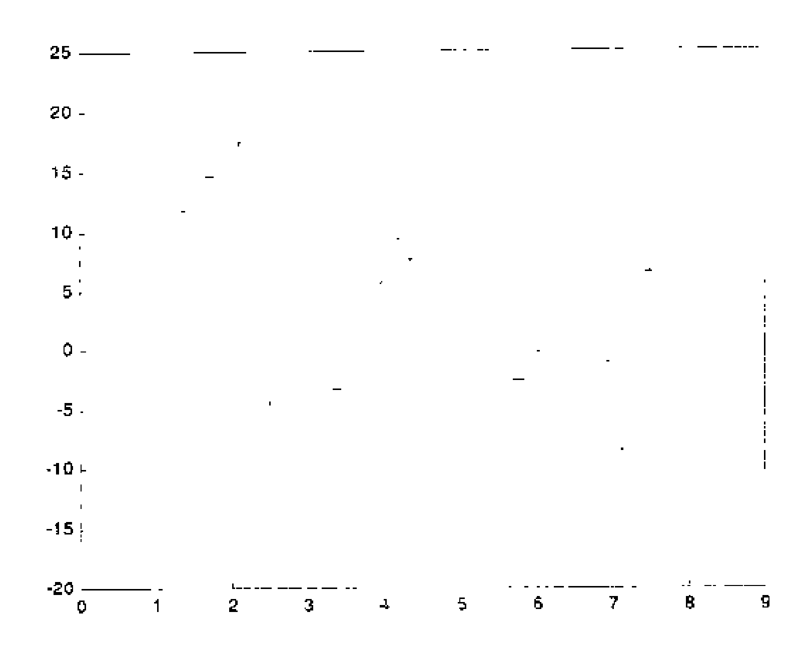



图 5-2 初始种群分布图

（2）经过一次遗传迭代后，寻优结果如图 5-3 所示。

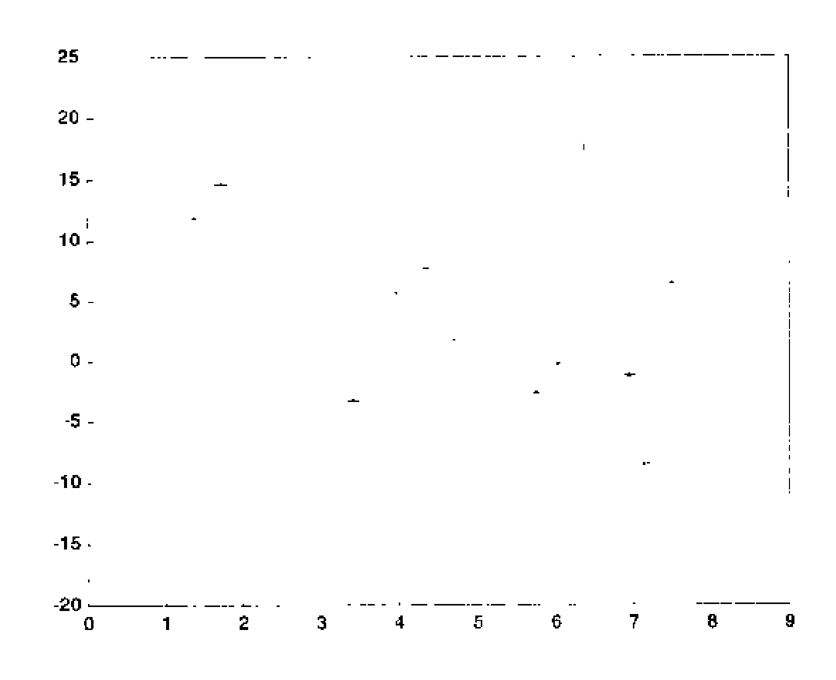



图 5-3 一次迭代后的寻优结果

图中 'o' 代表经过一次迭代后的个体分布，此时最优解为：

```
x =
          8.0217    20.1903
```

（3）经过 25 次迭代后，寻优结果如图 5-4 所示。

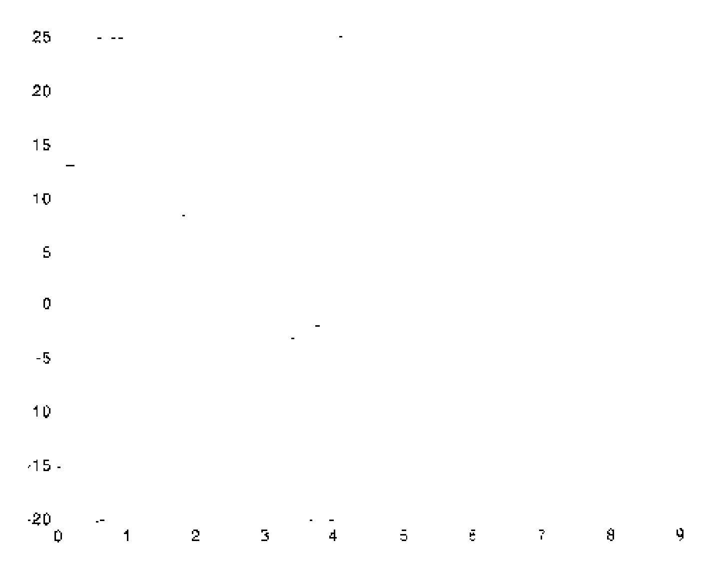



图 5-4 25 次迭代后的寻优结果

图中‘o’代表经过 25 次迭代后的个体分布，此时最优解为：

x = 7.8569　　24.8554

迭代过程如表 5-1 所示。

表 5-1 迭代过程

| 迭代次数 | 1 | 3 | 6 | 7 | 9 |
|---|---|---|---|---|---|
| 函数值 | 20.190259 | 20.551384 | 24.270734 | 24.421193 | 24.744628 |
| 迭代次数 | 11 | 15 | 16 | 25 | |
| 函数值 | 24.845763 | 24.855317 | 24.855361 | 24.855361 | |

即当 $x$=7. 8569 时，$f(x)$取最大值 24.8554。

（4）图 5-5 所示绘出了遗传算法的寻优性能。

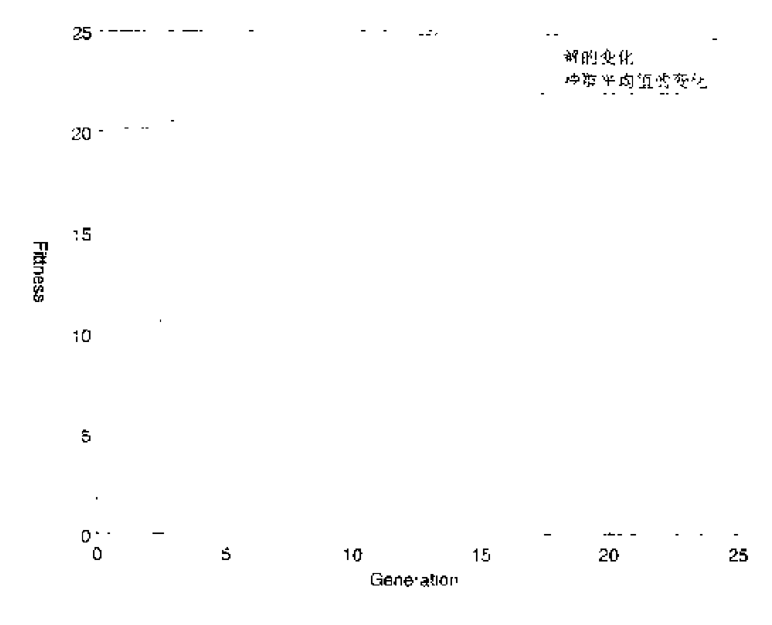



图 5-5 遗传算法的寻优性能

### 5.4.2 多维变量优化问题

【问题】

求 4 维 Corana 函数在区间[-10000 10000]内的极小值。

【分析】

Corana 函数实质上是一个 $n$ 维的抛物线，图 5-6 和图 5-7 分别表示二维 Corana 函数在 $x$、$y$ 平面内的切片形状。

【程序清单】

例程 5-7 是 $n$ 维 Corana 函数的 MATLAB 代码。

例程 5-7

```
function [eval] = corana(sol)
numv = size(sol,2);
x=sol(1:numv);
d0=[1 1000 10 100 1 10 100 1000 1 10];
d=d0(1:numv);
c=0.15;
s=.2*ones(1,numv);
t=0.05*ones(1,numv);
bk = s.*(round(x./s));
dev= (abs(bk-x)<t) & (bk~=0);
z=c*((bk+sign(bk).*t).^2).*d;
y=x.^2.*d;
eval = sum((dev.*z) + ((~dev).*y));
```

例程 5-8 是二维 Corana 函数的 MATLAB 示例代码，并做出其分别在 $x$、$y$ 平面内的切片图形。

例程 5-8

```
i=0;
a=-0.5:0.02:0.5;
for x=a
  i=i+1; j=0;
  for y=a
    j=j+1;
    z(i,j)=corana([x y]);
  end
end
clf
%x 平面内的图形
plot(z(:,1))
```

```
xlabel('x')
%x 的范围为[250.0-250.25]
clf
%y 平面内的图形
plot(z(1,:))
xlabel('y')
%y 的范围为[0-250]
%二维图形
mesh(a,a,z);
view(30,60);
grid;
```

图 5-6 x 平面上的形状

图 5-7 y 平面上的形状

例程 5-9 是遗传算法求解 4 维 Corana 函数极小值的 MATLAB 代码

**例程 5-9**

```
function [sol,eval] = coranaMin(sol,options)
numVar=size(sol,2)-1;
eval = corana(sol(1:numVar));
eval = -eval;
%设置变量的边界

bounds = ones(4,1)*[-10000 10000];
%遗传算法优化
[p,endPop,bestSols,trace]=ga(bounds,'coranaMin');
%性能跟踪
plot(trace(:,1),trace(:,3),'y-')
hold on
plot(trace(:,1),trace(:,2),'r-')
xlabel('Generation');
ylabel('Fittness');
legend('解的变化','种群平均值的变化');
```

【结果输出】

（1）最优解：

p1=-33.7847

p2= 0.1716

p3=-0.4250

p4=0.2218

（2）此时的适值，即极小值为

eval（p1，p2，p3，p4）=-182.4896

图 5-8 是二维 Corana 函数形状。图 5-9 是遗传算法寻优性能的跟踪图。

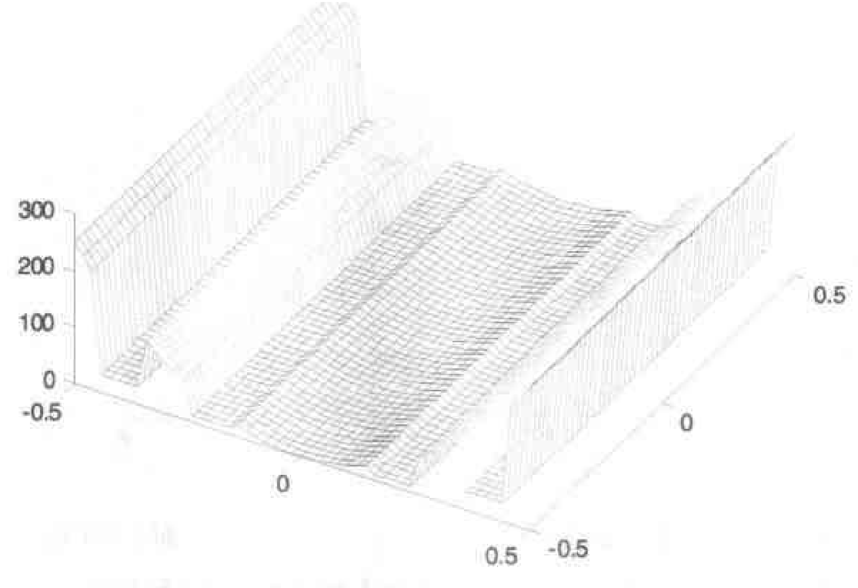



图 5-8　二维 Corana 函数形状

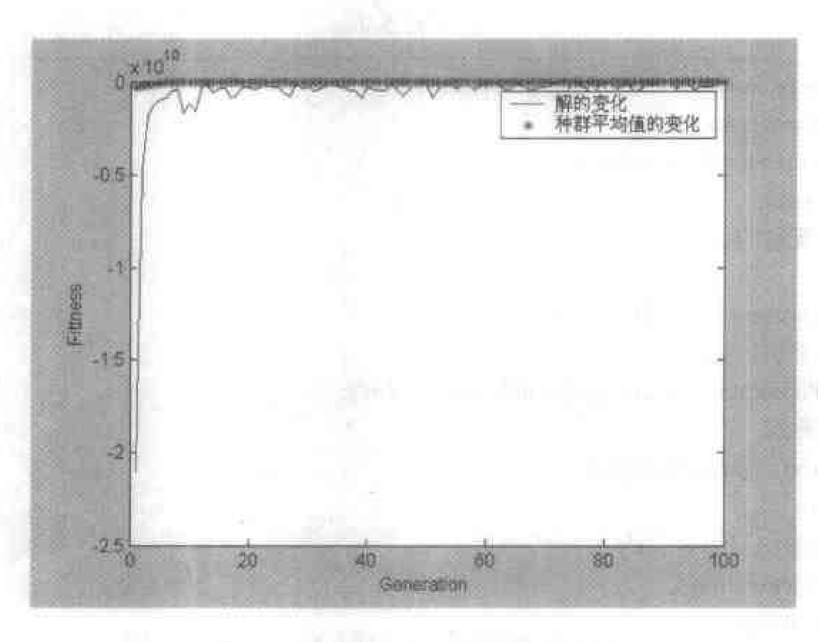



图 5-9　遗传算法的寻优性能跟踪图

# 5.5 遗传优化算法的工程应用

最优化在运筹学和管理科学中起着核心作用。最优化通常是极大或极小某个多变量的函数并满足一些等式和（或）不等式约束。最优化技术对社会的影响日益增加，应用的种类和数量快速增加，丝毫没有减缓的趋势。然而，许多工业工程设计问题性质十分复杂，用传统的优化方法很难解决。近年来，遗传算法作为一种全新的优化方法，以其巨大的潜力受到人们的普遍关注。本节将讨论遗传算法在无约束规划、非线性规划、可靠性优化、车间布局优化、参数优化和控制系统优化中的应用

## 5.5.1 遗传算法在无约束优化中的应用

### 1．基本理论

无约束优化针对的是没有任何约束前提下的极大或极小化某个函数的问题。一般来说，无约束优化问题可用如下数学公式描述：

$$\min \quad f(\boldsymbol{x}), \quad \boldsymbol{x}\in\Omega$$

其中，$f$ 是实值函数，可行域 $\Omega$ 是 $E^n$ 的子集。就可行域而言，当 $\Omega=E^n$ 时，问题是完全无约束的。

对点 $\boldsymbol{x}^*\in\Omega$，如果存在 $\varepsilon>0$，使得所有 $\boldsymbol{x}\in\Omega$ 与 $\boldsymbol{x}^*$ 的距离不大于 $\varepsilon$ 的点满足 $f(\boldsymbol{x})\geqslant f(\boldsymbol{x}^*)$，则称 $\boldsymbol{x}^*$ 是 $f$ 在 $\Omega$ 上的局部最优点；如果 $f(\boldsymbol{x})\geqslant f(\boldsymbol{x}^*)$ 对所有 $\boldsymbol{x}\in\Omega$ 均成立，则称 $\boldsymbol{x}^*$ 是 $f$ 在 $\Omega$ 上的全局最优点。

利用局部极小点的一阶必要条件，求函数极值的问题往往转化为求解 $\nabla f(\boldsymbol{x})=0$，即求 $\boldsymbol{x}$ 使满足（式 5.1）中

$$\begin{cases}\dfrac{\partial f(\boldsymbol{x})}{\partial(x_1)}=0\\ \quad\vdots\\ \dfrac{\partial f(\boldsymbol{x})}{\partial x_n}=0\end{cases} \tag{式 5.1}$$

问题。

$F$ 在点 $\boldsymbol{x}$ 的 Hessian 矩阵记为 $\nabla^2 f(\boldsymbol{x})$：

$$\nabla^2 f(\boldsymbol{x}) = [\frac{\partial^2 f(\boldsymbol{x})}{\partial x_i x_j}], \quad i, j = 1, \cdots, n$$

虽然绝大多数实际优化问题都有必须满足的约束条件，但无约束优化问题的研究是约束优化问题的基础。下面具体说明遗传算法在无约束优化问题中的应用。

2．实例分析

Ackley 函数是指数函数叠加上适度放大的余弦波再经调制而得到的连续型实验函数，如（式 5.2）所示：

$$f(\boldsymbol{x}) = -20 * e^{-0.2*\sqrt{\frac{1}{n}\sum_{j=1}^{n} x_j^2}} - e^{\frac{1}{n}\sum_{j=1}^{n}\cos(2\pi x_j)} + 22.71282 \qquad \text{（式 5.2）}$$

$n$=2 时，形状如图 5-6 所示。

例程 5-10 是绘制图 5-10 的 MATLAB 代码。

例程 5-10

```
[x1,x2] = meshgrid(-5:0.1:5);
f=-20*exp(-0.2*sqrt(0.5*(x1.^2+x2.^2)))-exp(0.5*(cos(2*pi*x1)+cos(2*pi*x2)))+22.71282;
mesh(x1,x2,f);
xlabel('x1');
ylabel('x2');
zlabel('f(x1,x2)');
```

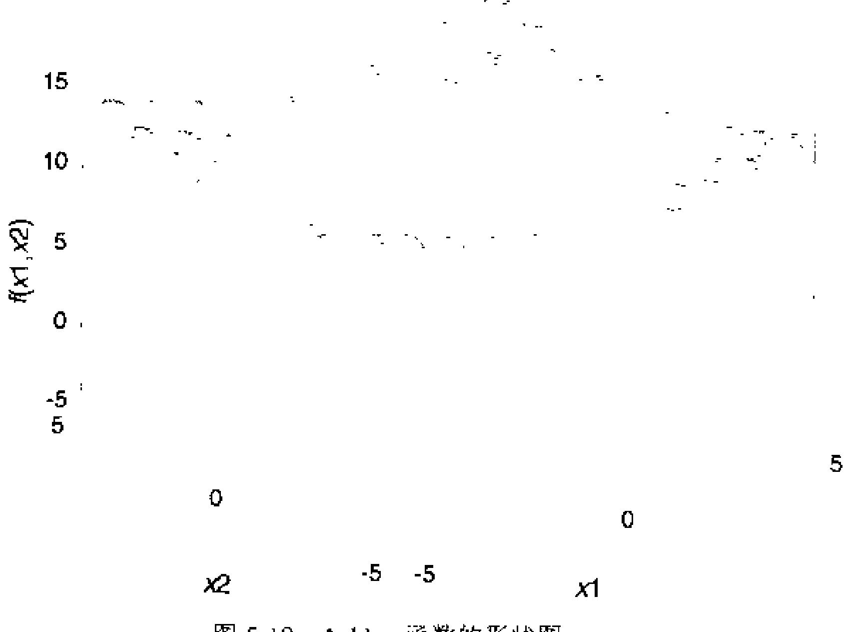



图 5-10 Ackley 函数的形状图

【问题】

在 $-5 \leqslant x_j \leqslant 5$，$j=1,2$ 区间内，求解 $\min f(x_1, x_2)$。

Ackley 指出，这个函数的搜索十分复杂，因为一个严格的局部最优算法在爬山过程中不可避免地要落入局部最优的陷阱；而扫描较大领域就能越过干扰的山谷，逐步达到较好的最优点。所以求解 Ackley 函数的最小值是遗传优化算法应用的一个有力的例证。

【分析】

为极小化 Ackley 函数，采用遗传算法来实现。按照遗传算法的基本步骤进行种群初始化、适应度估计、选择、交叉和变异运算，编码采用实数。

遗传算法的参数设置如下：

种群大小 pop_size=10

最大代数 gen_max=1000

变异率 $p_m$=0.1

交叉率 $p_c$=0.3

初始种群是在[-5 5]区间内随机产生的，如表 5-2 所示。

表 5-2　初始种群值

| | x1 | x2 |
|---|---|---|
| p1 | 4.5013 | 1.1543 |
| p2 | -2.6886 | 2.9194 |
| p3 | 1.0684 | 4.2181 |
| p4 | -0.1402 | 2.3821 |
| p5 | 3.9130 | -3.2373 |
| p6 | 2.6210 | -0.9429 |
| p7 | -0.4353 | 4.3547 |
| p8 | -4.8150 | 4.1690 |
| p9 | 3.2141 | -0.8973 |
| p10 | -0.5530 | 3.9365 |

相应的适值为：

eval（p1）=f（4.5013, 1.1543）=11.5418

eval（p2）=f（-2.6886, 2.9194）=10.0203

eval（p3）=f（1.0684, 4.2181）=10.1639

eval（p4）=f（-0.1402, 2.3821）=7.4906

eval（p5）=f（3.9130, -3.2373）=11.3654

eval（p6）=f（2.6210, -0.9429）=8.1130

eval（p7）=f（-0.4353, 4.3547）=11.4769

eval（p8）=f（-4.8150, 4.1690）=13.0315

eval（p9）=f（3.2141, -0.8973）=8.5692

eval（p10）=f（-0.5530, 3.9365）=10.3252

【程序清单】

例程 5-11 是 Ackley 函数的 MATLAB 代码。

例程 5-11

```
function [eval]=griewangk(sol)
numv = size(sol,2);
x=sol(1:numv);
eval=-20*exp(-0.2*sqrt(sum(x.^2)/numv))-exp(sum(cos(2*pi*x))/numv)+22.71282;
```

例程 5-12 是计算 Ackley 函数适值的 MATLAB 代码。

例程 5-12

```
function [sol,eval]=Ackleymin(sol,options)
numv = size(sol,2)-1;
x=sol(1:numv);
eval=Ackley(x);
eval=-eval;
```

例程 5-13 是遗传算法求解的 MATLAB 代码。

例程 5-13

```
%维数 n=2
%设置参数边界
bounds = ones(2,1)*[-5    5];
%遗传算法优化
[p,endPop,bestSols,trace]=ga(bounds,'Ackleymin');

%性能跟踪
plot(trace(:,1),trace(:,3),'b-')
hold on
plot(trace(:,1),trace(:,2),'r-')
xlabel('Generation');
ylabel('Fittness');
legend('解的变化','种群平均值的变化');
```

【结果输出】

（1）最优解为：p =

0.0000
0.0000

（2）此时 Ackley 函数的适值，即极小值为：

Ackley(p)=0.0055

在理论上，最优解应为 p=0 0 0，极小值为 0。显然，遗传算法有效地解决了 Ackley 函数的极小化问题。

图 5-11 是遗传算法寻优性能的跟踪图。

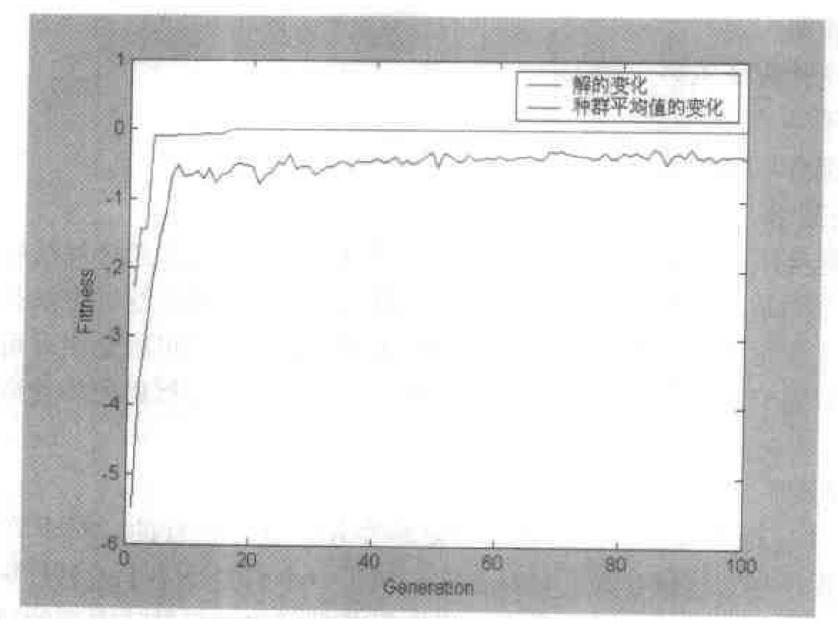



图 5-11 遗传算法寻优性能的跟踪图

## 5.5.2 遗传算法在非线性规划中的应用

### 1. 基本原理

非线性规划是在存在等式或不等式约束的前提下最优化某个目标函数的问题。由于许多实际问题不能成功地表达为线性规划模型，因此非线性规划对于工程、数学和运筹学的各个领域都是极其重要的工具。一般非线性规划可描述如下：

$$\max f(x)$$

满足：

$$g_i(x)\leqslant 0,\quad i=1,\cdots,m$$
$$h_i(x)=0,\quad i=m+1,\cdots,n$$
$$x\in\Omega$$

上述问题要求在变量 $x_1,\cdots,x_n$ 满足约束的同时极小化函数 $f$。函数 $f$ 通常称之为目标函数。约束 $g_i(x)\leqslant 0$ 称之为不等式约束，$h_i(x)=0$ 称之为等式约束。与线性规划不同，传统的非线性规划方法十分复杂且效率不高。在过去几年里，遗传算法在非线性规划中应用不断增加。

由于对染色体做遗传运算通常获得不可行的后代，因此运用遗传算法解非线性规划的核心是如何满足约束的问题。近年来已经提出了几种用遗传算法满足约束的技术，这些技术大致可分为以下几类：

- 拒绝策略
- 修复策略
- 改进遗传算子策略
- 惩罚策略

各种策略都有不同的优点和缺点

（1）拒绝策略

拒绝策略抛弃所有进化过程中产生的不可行的染色体，这是遗传算法中普遍的做法。当可行的搜索空间是凸的且为整个搜索空间的适当的一部分时，这种方法是有效的。然而这样的条件是比较苛刻的。例如对许多约束优化问题初始种群可能是由非可行染色体构成的，这就需要对它们进行修补。对于某些系统，允许跨过不可行染色体使修复往往更能达到最优解。

（2）修复策略

修补染色体是对不可行染色体采用修复程序使之变为可行的。对于许多组合优化问题，构造修复程序相对比较容易。已经证明，对于一个有多个不连通可行集的约束组合优化问题，修复策略在速度和计算性能上都远胜过其他策略。但是该方法的缺点是它对问题本身的依赖性，对于每个具体问题必须设计专门的修复程序，而对某些问题，修复过程本身比原问题的求解更复杂。

（3）改进遗传算子策略

解决可行性问题的一个合理办法是设计针对问题的表达方式，以及专门的遗传算子来维持染色体的可行性。许多领域中的实际工作者采用专门的问题表达方式和遗传算子构成了非常成功的遗传算法，这已经是一个十分普遍的趋势，但是该方法遗传搜索受到了可行域的限制。

（4）惩罚策略

上面的三种策略的共同优点是都不会产生不可行解，缺点是无法考虑可行域外的点。对于约束严的问题，不可行解在种群中的比例很大，这样将搜索限制在可行域内就很难找到可行解。惩罚策略就是这类在遗传搜索中考虑不可行解的技术。

工程中常用的方法是惩罚策略。本质上它是通过惩罚不可行解将约束问题转化为无约束问题。在约束算法中，惩罚技术用来在每一代的种群中保持部分不可行解，使遗传搜索可以从可行域和不可行域两边来达到最优解。惩罚策略的关键问题是如何设计一个惩罚函数 $p(x)$，从而能有效地引导遗传搜索达到解空间的最好区域。不可行染色体和解空间可行部分的关系在惩罚不可行染色体中起了关键作用：不可行染色体的惩罚相应于某种测度下的不可行性的测量。

构造带有惩罚项的适值函数一般有两种，一种是采用加法形式：

$$val(x) = f(x) + p(x)$$

其中。$x$ 代表染色体；$f(x)$ 是问题的目标函数；$p(x)$ 是惩罚项。

对于极大化问题，则取：

$$\begin{cases} p(x)=0, & \text{若}x\text{可行} \\ p(x)>0, & \text{其他} \end{cases}$$

对于极小化问题，则取：

$$\begin{cases} p(x)=0, & \text{若}x\text{可行} \\ p(x)<0, & \text{其他} \end{cases}$$

另一种是采用乘法形式：

$$val(x)=f(x)*p(x)$$

此时，对于极大化问题，则取：

$$\begin{cases} p(x)=1, & \text{若}x\text{可行} \\ 0\leqslant p(x)<1, & \text{其他} \end{cases}$$

对于极小化问题，则取：

$$\begin{cases} p(x)=0, & \text{若}x\text{可行} \\ p(x)>1, & \text{其他} \end{cases}$$

2．**实例分析**

考虑如下问题

$$\min f(x)=(x_1-2)^2+(x_2-1)^2$$

$$s.t.\quad \begin{aligned} g_1(x)&=x_1-2x_2+1\geqslant 0 \\ g_2(x)&=\frac{x_1^2}{4}-x_2^2+1\geqslant 0 \end{aligned}$$

**【分析】**

采用 Homaifar、Qi 和 Lai 方法求解。

Homaifar 等考虑如下非线性规划问题：

$$\min f(x)$$

$$s.t.\quad g_i(x)\geqslant 0,\quad i=1,\cdots,m$$

取加法形式的适值函数：

$$val(x)=f(x)+p(x)$$

惩罚函数由两部分构成，可变乘法因子和违反约束乘法，其表达式如下：

$$p(x)=\begin{cases}0 & 若x可行\\ \sum_{i=1}^{m} r_i g_i(x) & 其他\end{cases}$$

其中 $r_i$ 是约束 $i$ 的可变惩罚系数.

选择二进制编码，种群中的个体数目为 100，实数编码，交叉概率为 0.95，变异概率为 0.08。

**【程序清单】**

例程 5-14 是计算目标函数适值的 MATLAB 代码。

例程 5-14

```
function [sol,eval]=f552(sol,options)
x1=sol(1);
x2=sol(2);
r1=0.1;
r2=0.8;
%约束条件
g1=x1-2*x2+1;
g2=x1.^2/4-x2.^2+1;
%加惩罚项的适值
if (g1>=0)&(g2>=0)
  eval=(x1-2).^2+(x2-1).^2;
else
  eval=(x1-2).^2+(x2-1).^2+r1*g1+r2*g2;
  eval=-eval;
end
```

例程 5-15 是遗传算法求解的 MATLAB 代码。

例程 5-15

```
%n=2
%设置参数边界
bounds = ones(2,1)*[-1 1];

%遗传算法优化
[p,endPop,bestSols,trace]=ga(bounds,'rosenbrockMin');

%性能跟踪
plot(trace(:,1),trace(:,3),'b-')
hold on
plot(trace(:,1),trace(:,2),'r-')
xlabel('Generation');
ylabel('Fittness');
legend('解的变化','种群平均值的变化');
```



**【结果输出】**

（1）最优解为：p =

1.000

1.000

（2）此时极小值为：

eval(p)=1

$g_1(p)=0$

$g_2(p)=0.25$

显然最优解满足约束条件。

图 5-12 是遗传算法寻优的跟踪图。

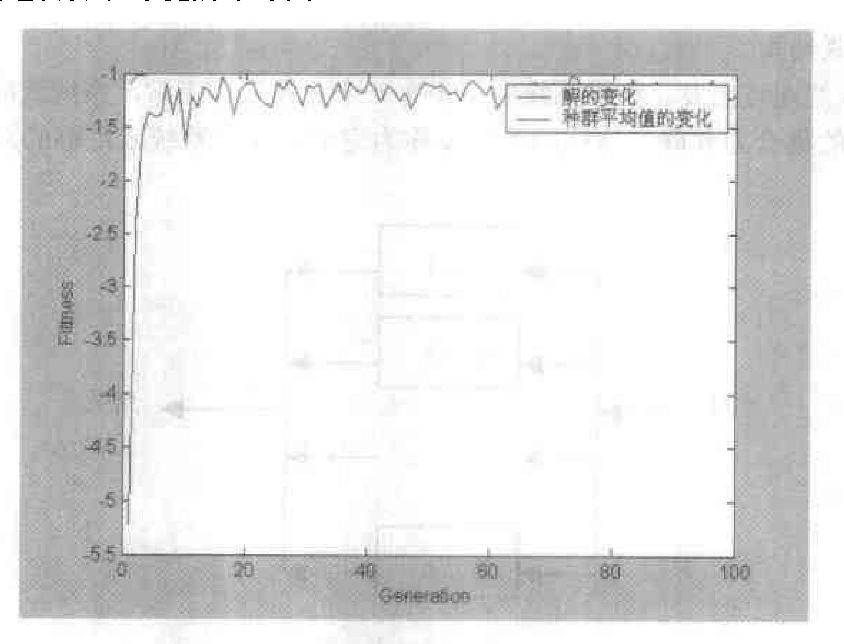



图 5-12　遗传算法寻优的跟踪图

### 5.5.3　遗传算法在可靠性优化中的应用

#### 1. 简介

从广义上讲，可靠性是一个系统性能的度量。随着系统复杂性程度的增加，系统的非可靠性以费用、费力程度和寿命等方面的问题表现出来。因而对系统可靠性的评价、产品和系统可靠性的改善就显得越来越重要。

研究可靠性问题的主要目标之一是寻求最好的方法提高系统的可靠性，可靠性优化有助于可靠性工程师实现目标。已经有许多改进系统可靠性的方法，但实践证明比较好的是冗余方法。冗余优化模型假设系统由 $n$ 个独立子系统组成，系统在串并联中均有冗余的元件，并有替换的设计方案。优化主要集中于冗余元件的最优分配和比较设计的最优选择，以满足系统的需求。系统冗余可靠性优化方法较多，但在用于大规模非线性优化问题时，仅有少数算法被证明是有效的，没有哪一种算法被证明比其他的算法更优越。

复杂系统常常分解为由单元、子系统或用于可靠性分析的元件组成的功能实体，连接的方式包括串联和并联两种。

（1）串联结构

从可靠性的观点看，如果系统正常工作取决于组成系统的所有元件都正常工作，则称 $n$ 个元件的集合为串联。换句话说，只有组成系统的所有元件工作正常，系统才能工作正常，这样的系统被称为非冗余系统，如图 5-13 所示。

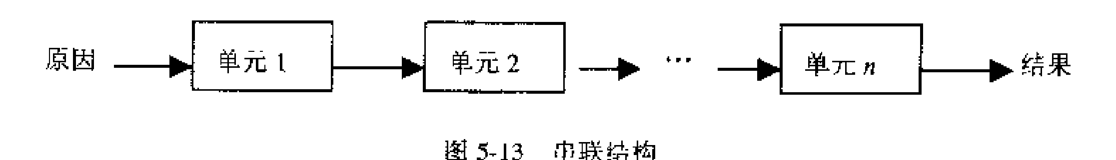



图 5-13 串联结构

（2）并联结构

从可靠性的观点出发，如果系统中至少有一个元件工作正常，系统就能工作正常，称这 $n$ 个元件的集合为并联，这样的系统被称为完全的冗余系统或足够的冗余系统，如图 5-14 所示。

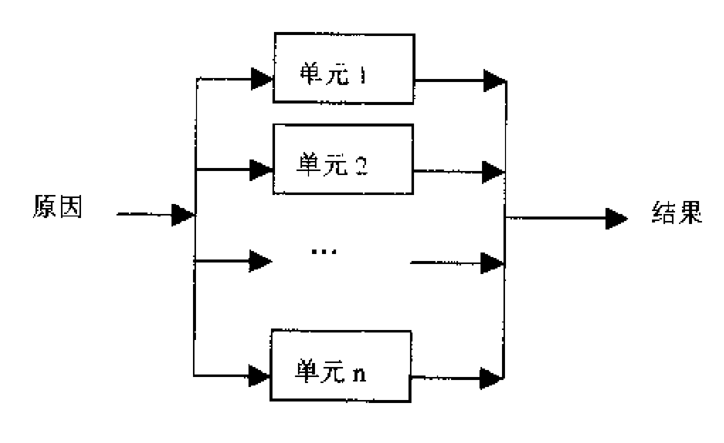



图 5-14 并联结构

### 2. 可靠性优化的简单遗传算法

**【问题】**

Gen、Ida 和 Yokata 首次提出了求解可靠性优化的简单遗传算法。问题的目标是在受限于 A 类失效（整个系统满足失效条件时）的子系统中具有并联冗余单元的三个非线性约束条件下使系统可靠性达到最大，表述为如下数学模型：

$$\max R(m)=\prod_{i=1}^{3}(1-(1-(1-q_{i1})^{m_i+1})-\sum_{n=2}^{4}(q_{in})^{m_i+1})$$

使得满足：



$$G_1(m)=(m_1+3)^2+m_2^2+m_3^2\geqslant 51$$

$$G_2(m)=20\sum_{i=1}^{3}(m_i+e^{-m_i})\geqslant 120$$

$$G_3(m)=20\sum_{i=1}^{3}m_ie^{-\frac{m_i}{4}}\geqslant 65$$

$$1\leqslant m_1\leqslant 4,\quad 1\leqslant m_2,\quad m_3\leqslant 7,\quad m_i\geqslant 0,\ \text{整数},\quad i=1,2,3$$

其中，$m=[m_1,m_2,m_3]$，对于子系统受限于一种 O 类失效（$h_i=1$）和三种 A 类失效的四种失效模式（$s_i=4$），$i$=1,2,3，每个子系统的失效概率如表 5-3 所示。

表 5-3　每个子系统的失效模式和失效概率

| 子系统 $i$ | 失效模式（$s_i$=4,$h_i$=1） | 失效概率（$q_{ik}$） |
|---|---|---|
| 1 | O | 0.01 |
|  | A | 0.05 |
|  | A | 0.10 |
|  | A | 0.18 |
| 2 | O | 0.08 |
|  | A | 0.02 |
|  | A | 0.15 |
|  | A | 0.12 |
| 3 | O | 0.04 |
|  | A | 0.05 |
|  | A | 0.20 |
|  | A | 0.10 |

**【分析】**

（1）染色体表达

变量 $m_i$ 的整数值用二进制串表达，串的长度取决于冗余单元的上界 $u_i$。本例中，每个子系统冗余单元的上界 $u_1$=4,$u_2$=7,$u_3$=7，因此决策变量 $m_i$ 需要 3 位二进制位，这样表达 $m$ 总共需要 9 位。

（2）初始种群

每个染色体是一个 9 位的二进制串，随机产生的染色体可能由于违背约束或（和）超过上界而不可行，因此需要通过可行性检查以保证所有的染色体是可行的。

（3）染色体的评估

对于每个可行的染色体，适值赋予目标函数值 $R(m)$，而对于每个不可行的染色体，给予一个很大的惩罚，即：

$$eval(p_k)=\begin{cases}R(m), & 若m可行\\ -M, & 其他\end{cases}$$

其中 $p_k$ 代表第 $k$ 个染色体，$M$ 是一个很大的正整数。

遗传算法参数设置：选择二进制编码，种群中的个体数目为 10，二进制编码序列的长度为 9，交叉概率为 0.4，变异概率为 0.1。

**【程序清单】**

例程 5-16 是适值函数的 MATLAB 代码。

例程 5-16

```
function [sol,eval]=f553(sol,options)
m(1)=sol(1);
m(2)=sol(2);
m(3)=sol(3);
%失效概率矩阵
q=[0.01 0.05 0.10 0.18;
0.08 0.02 0.15 0.12;
0.04 0.05 0.20 0.10];
%约束条件
g1=51-(m(1)+3).^2+m(2).^2+m(3).^2;
g2=20*sum(m+exp(-m))-120;
g3=20*sum(m.*exp(-m/4))-65;
%计算加惩罚项的适值
if ((g1>=0)&(g2>=0)&(g3>=0))
  multi=1;
  for i=1:3
     summ=0;
     for j=2:4
        summ=summ+q(i,j).^(m(i)+1);
     end
     multi=multi*(1-(1-(1-q(i,1)).^(m(i)+1))-summ);
  end
  eval=multi;
else
%取 M=500
  eval=-500;
end
```

例程 5-17 是求解 MATLAB 代码。

例程 5-17

```
%维数 n=3
%设置参数边界
bounds =[1 4:1 7;1 7];
```

```
%遗传算法优化
%. 进制编码
options=[1e-6 2];
%initPop=initializega(10,bounds,'f553',[],options);
%[m endPop] = ga(bounds,'f553',[],initPop,[1e-6 1 1],'maxGenTerm',100,'normGeomSelect',...
[0.08],['arithXover'],[2], 'nonUnifMutation',[2 5 3]);

[p,endPop,bestSols,trace]=ga(bounds,'f553');

%性能跟踪
plot(trace(:,1),trace(:,3),'b-');
hold on
plot(trace(:,1),trace(:,2),'r-');
xlabel('Generation');
ylabel('Fittness');
legend('解的变化','种群平均值的变化');
```

**【结果输出】**

（1）最优解：

m= 3 1 2

（2）此时的适值为

eval（m）=0.6607

图 5-15 是遗传算法寻优性能的跟踪图。

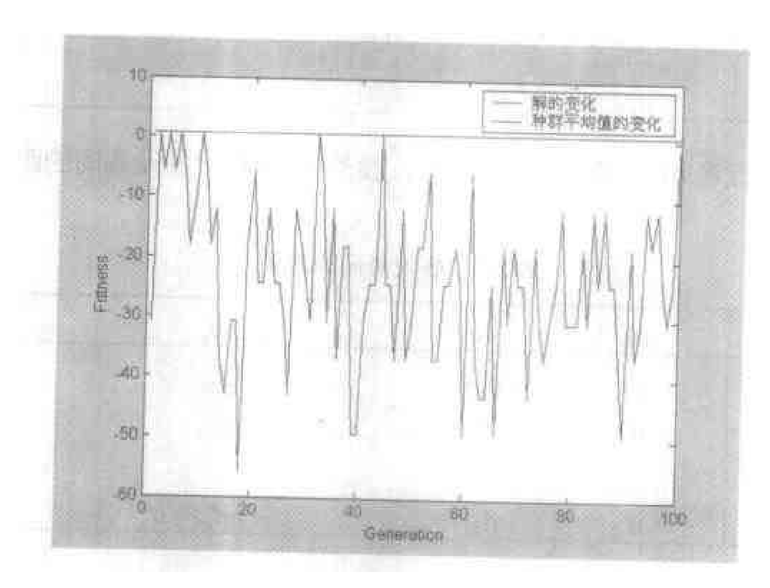



图 5-15 遗传算法寻优性能的跟踪图

## 5.5.4 遗传算法在车间布局优化中的应用

1．引言

迄今为止，制造技术取得了重大发展，在世界范围内实现了大量的柔性制造系统，制

造系统物理布局优化设计是在系统设计初期必须解决的重要问题之一。随着经济形势的发展，制造业的竞争逐步从规模竞争、质量竞争逐步转向速度竞争，缩短从订货到交货的周期是赢得市场的首要因素。许多与此有关的新技术得到迅速的发展和应用。据统计，在一个产品中，等待的时间占的比例达到 90%～95%，真正用于产品加工的实践所占的比例很小，其中车间内物流流动的时间是影响效率的主要因素之一，因此需要改变车间内设备的布局，使设备尽可能按照产品的工艺流程顺序布置，可有效地减少搬运时间和降低生产成本。几十年来，设备布局设计一直是跨学科的研究课题。从数学的角度来看，设备布局优化问题属于非线性规划问题，由于约束条件较多，对较多设备的布局优化问题求解时，将很难找到一种最优解。近些年来，用遗传算法解决设备布局设计的趋势迅速增加，例如将遗传算法用于有形状约束的不等区域设备布局问题、解决车间布局设计的分布式遗传算法等。本节将举例说明遗传算法在车间设备布局优化设计中的应用。

2．车间设备布局优化问题

假设车间中共有 $n$ 个机床设备 $M=(m_1,\cdots,m_n)$，工件在各设备间的传输次数如表 5-4 所示。

表 5-4　工件在各设备间的传输次数

| | $m_1$ | $m_2$ | … | $m_n$ |
|---|---|---|---|---|
| $m_1$ | $a_{11}$ | $a_{12}$ | … | $a_{1n}$ |
| $m_2$ | $a_{21}$ | $a_{22}$ | … | $a_{2n}$ |
| | … | … | … | … |
| $m_n$ | $a_{n1}$ | $a_{n2}$ | … | $a_{nn}$ |

其中 $a_{ij}$ 表示由设备 $m_i$ 传输至 $m_j$ 的次数，显然 $a_{ii}=0$。各设备间的距离如表 5-5 所示。

表 5-5　各设备间的距离

| | $m_1$ | $m_2$ | … | $m_n$ |
|---|---|---|---|---|
| $m_1$ | $d_{11}$ | $d_{12}$ | … | $d_{1n}$ |
| $m_2$ | $d_{21}$ | $d_{22}$ | … | $d_{2n}$ |
| | … | … | … | … |
| $m_n$ | $d_{n1}$ | $d_{n2}$ | … | $d_{nn}$ |

其中 $d_{ij}$ 表示由设备 $m_i$ 至 $m_j$ 的距离，显然 $d_{ii}=0$。

**【分析】**

车间设备布局最优设计应使得各设备间的运输距离最短，即

$$\min\sum_{i=1}^{n}\sum_{j=1}^{n}a_{ij}d_{ij}$$

另外，各设备在车间中均需占有一定的面积，设备间还需为操作人员留有一定的活动空间，因此在车间设备布局设计中还需要考虑一些约束条件。设备 $i$、$j$ 间的相对位置图如图 5-16 所示。

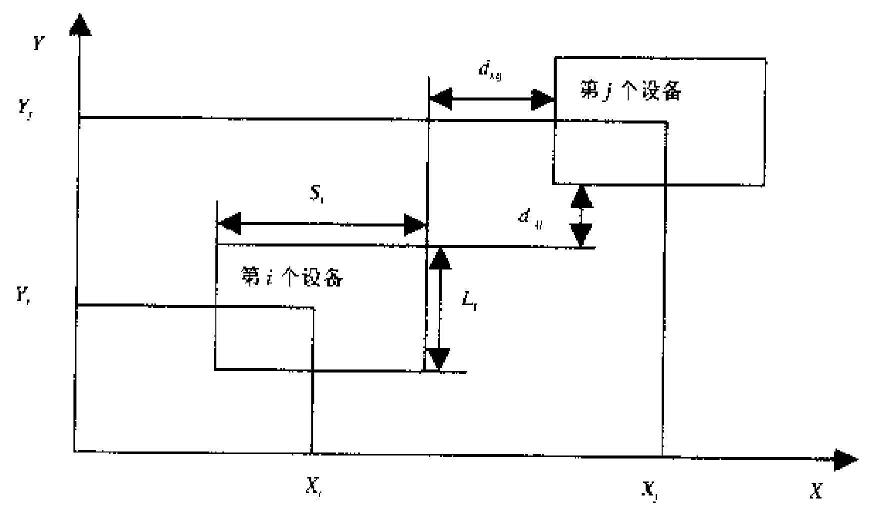



图 5-16　设备 $i$、$j$ 间的相对位置图

其中 $d_{xij}, d_{yij}$ 分别表示设备在 $X$ 和 $Y$ 方向的最小间距，$S_i, L_i$ 分别表示设备 $i$ 在 $X$ 和 $Y$ 方向的宽度。

实现车间布局的优化设计，所需要的约束条件包括：

（1）间距约束：设备间应保持一定的间距，即

$$|X_i - X_j| \geqslant \frac{S_i + S_j}{2} + d_{xij}$$

$$|Y_i - Y_j| \geqslant \frac{L_i + L_j}{2} + d_{yij}$$

（2）边界约束：设备在 $X$、$Y$ 方向的布置不应超过车间的长度尺寸，即

$$\sum_{i=1}^{n-1}(|X_i - X_{i+1}| + \frac{S_i + S_{i+1}}{2}) \leqslant h$$

$$\sum_{i=1}^{n-1}(|Y_i - Y_{i+1}| + \frac{L_i + L_{i+1}}{2}) \leqslant g$$

，$h$、$g$ 分别表示车间在 $X$、$Y$ 方向上的宽度

- 例程参数设置

车间大小（5m×5m），放置 5 台设备，工件在各设备间的传输次数如表 5-6 所示。

表 5-6　工件在各设备间的传输次数（单位：次）

| | $m_1$ | $m_2$ | $m_3$ | $m_4$ | $m_5$ |
|---|---|---|---|---|---|
| $m_1$ | 0 | 572 | 1559 | 157 | 0 |
| $m_2$ | 15 | 0 | 26 | 0 | 2 |

（续表）

| | $m_1$ | $m_2$ | $m_3$ | $m_4$ | $m_5$ |
|---|---|---|---|---|---|
| $m_3$ | 6 | 54 | 0 | 64 | 36 |
| $m_4$ | 0 | 14 | 37 | 0 | 38 |
| $m_5$ | 7 | 4 | 0 | 22 | 0 |

设备的尺寸如表 5-7 所示.

表 5-7　设备的尺寸（单位：m）

| $m_1$ | $m_2$ | $m_3$ | $m_4$ | $m_5$ |
|---|---|---|---|---|
| 3×3 | 1×0.8 | 1.5×0.8 | 1.5×0.8 | 0.8×0.7 |

$$d_{xij}=0.8, d_{yij}=0.5$$

- 遗传算法参数设置

选择实数编码，种群中的个体数目为 10，最大代数 100，交叉概率为 0.9，变异概率为 0.04。

【程序清单】

例程 5-18 是适值函数的 MATLAB 代码.

例程 5-18

```
function [sol,eval]=f554(sol,options)
x(1:5)=sol(1:5);
y(1:5)=sol(6:10);
%传输次数矩阵
a=[0   572 1559 157 0;
   15 0    26   0    2;
   6  54   0    64   36;
   0  14   37   0    38;
   7  4    0    22   0];
%x 方向尺寸向量
S=[3 1 1.5 1.5 0.8];
%y 方向尺寸向量
L=[3 0.8 0.8 0.8 0.7];
%设备 x,y 在 x 方向上的最小间距
dxijmin=0.8;
%设备 x,y 在 y 方向上的最小间距
dyijmin=0.5;
%车间尺寸
H=5;
G=5;

for i=1:5
  for j=1:5
```

```
            delta1(i,j)=abs(x(i)-x(j))-(S(i)+S(j))/2-dxijmin;
            delta2(i,j)=abs(y(i)-y(j))-(L(i)+L(j))/2-dyijmin;
            %设备 i，j 之间的距离
            d(i,j)=sqrt((x(i)-x(j)).^2+(y(i)-y(j)).^2);
        end
    end

    %约束 1
    delta11=min(min(delta1));
    %约束 2
    delta22=min(min(delta2));
    summ1=0;
    for i=1:4
    summ1=summ1+abs(abs(x(i)-x(i+1))+(S(i)+S(i+1))/2);
    end
    %约束 3
    summ11=H-summ1;

    summ2=0;
    for i=1:4
      summ2=summ2+abs(abs(y(i)-y(i+1))+(L(i)+L(i+1))/2);
    end
    %约束 4
    summ22=G-summ2;

    if ((delta11>=0)&(delta22>=0)&(summ11>=0)&(summ22>=0))
      fsum=0;
      for i=1:5
          for j=1:5
            fsum=fsum+a(i,j)*d(i,j);
          end
      end
      eval=fsum;
    else
      %惩罚项
      eval=-500;
    end

    eval=-eval;
```

例程 5-19 是遗传算法求解的 MATLAB 代码。

**例程** 5-19

```
%维数 n=5
%车间尺寸
H=5;
G=5;
```

```
%x 方向尺寸向量
S=[3 1 1.5 1.5 0.8];
smin=min(S)/2;
%y 方向尺寸向量
L=[3 0.8 0.8 0.8 0.7];
lmin=min(L)/2;
%设备 x,y 在 x 方向上的最小间距
dxijmin=0.8;
%设备 x,y 在 y 方向上的最小间距
dyijmin=0.5;
%设置参数边界
bounds =[smin H;smin H;smin H;smin H;smin H;
lmin G;lmin G;lmin G;lmin G;lmin G];
% 生成初始种群，大小为 10，且满足约束条件
flag=0;
while flag<11
  init=initializega(1,bounds,'f554');
  x(1:5)=init(1:5);
  y(1:5)=init(6:10);

  for i=1:5
      for j=1:5
        delta1(i,j)=abs(x(i)-x(j))-(S(i)+S(j))/2-dxijmin;
        delta2(i,j)=abs(y(i)-y(j))-(L(i)+L(j))/2-dyijmin;
      end
  end

  %约束 1
  delta11=min(min(delta1));
  %约束 2
  delta22=min(min(delta2));

  summ1=0;
  for i=1:4
    summ1=summ1+abs(x(i)-x(i+1))+(S(i)+S(i+1))/2;
  end
  %约束 3
  summ11=H-summ1;

  summ2=0;
  for i=1:4
    summ2=summ2+abs(y(i)-y(i+1))+(L(i)+L(i+1))/2;
  end
  %约束 4
  summ22=G-summ2;
```

```
    if ((delta11>=0)&(delta22>=0)&(summ11>=0)&(summ22>=0))
        flag=flag+1;
        initPop(flag,:)=init;
    else
        continue;
    end
end
% 调用遗传函数
[p endPop bpop trace] = ga(bounds,'f554',[],initPop,[1e-5 1 1],'maxGenTerm',100,...
'normGeomSelect',[0.08],['arithXover'], [20], 'nonUnifMutation',[2 1 3]);
```

**【结果输出】**

最优解为：p =

```
0.8036
1.4812
2.8355
3.8879
3.3534
0.8207
0.4420
0.8821
0.9973
4.5129
```

即设备布局的位置分别为：（0.8036,0.8207）、（1.4812,0.4420）、（2.8355,0.8821）、（3.8879,0.9973）和（3.3534,4.5129）。

### 5.5.5 遗传算法在参数优化中的应用

1．De Jong 函数的极小化问题

**【问题】**

De Jong 函数是一个连续的、凸起的单峰函数，表达式如（式 5.3）所示，

$$f(x)=\sum_{i=1}^{n}x_i^2 \qquad \text{（式 5.3）}$$

求解：　　$\min f(x) \qquad -512 \leqslant x_i \leqslant 512$

二维的 De Jong 函数图形如图 5-17 所示。

例程 5-20 是绘制图 5-17 的 MATLAB 代码.

例程 5-20

```
[x1,x2] = meshgrid(-512:4:512);
```

```
f=x1.^2+x2.^2;
mesh(x1,x2,f);
xlabel('x1');
ylabel('x2');
zlabel('f(x1,x2)');
```

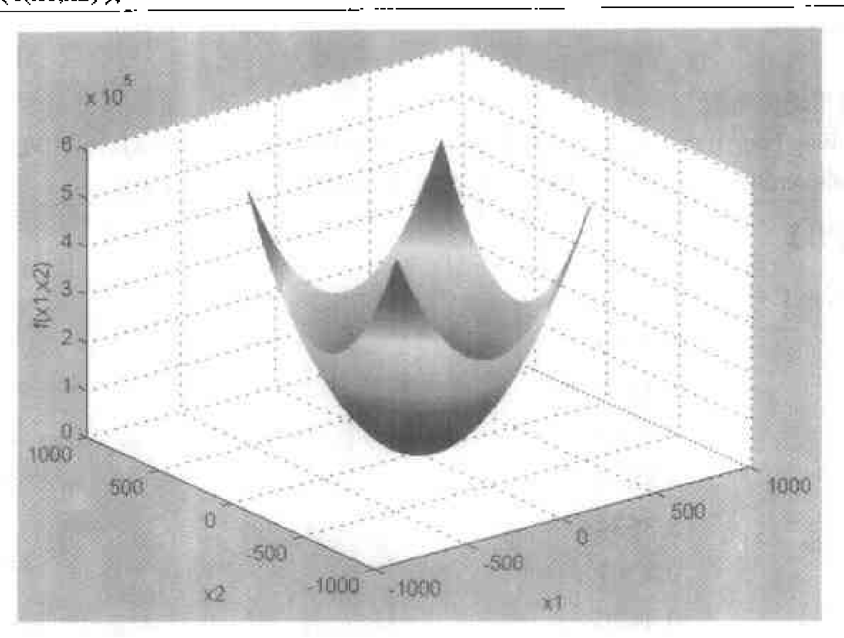



图 5-17　二维的 De Jong 函数图形

**【程序清单】**

例程 5-21 是 De Jong 函数的 MATLAB 代码。

**例程 5-21**

```
function [eval]=dejong(sol)
numv = size(sol,2);
x=sol(1:numv);
eval=sum(x.^2);
```

例程 5-22 是计算 De Jong 函数适值的 MATLAB 代码。

**例程 5-22**

```
function [sol,eval]=dejongmin(sol,options)
numv = size(sol,2)-1;
x=sol(1:numv);
eval=dejong(x);
eval=-eval;
```

例程 5-23 是遗传算法求解的 MATLAB 代码。

**例程 5-23**

```
%维数 n=3
%设置参数边界
```



```
bounds = ones(3,1)*[-512 512];
%遗传算法优化
[p,endPop,bestSols,trace]=ga(bounds,'dejongmin');

%性能跟踪
plot(trace(:,1),trace(:,3),'b-')
hold on
plot(trace(:,1),trace(:,2),'r-')
xlabel('Generation');
ylabel('Fittness');
legend('解的变化','种群平均值的变化');
```

【结果输出】

（1）最优解为：p =

1.0e-003 *

0.2500

0.1567

0.0500

（2）此时 De Jong 函数的适值，即极小值为：

dejong(p)=1.0e-003 *　　-0.0001

而在理论上，最优解应为：p=0 0 0，极小值为 0。显然，遗传算法有效地解决了 De Jong 函数的极小化问题。

图 5-18 是遗传算法寻优性能的跟踪图。

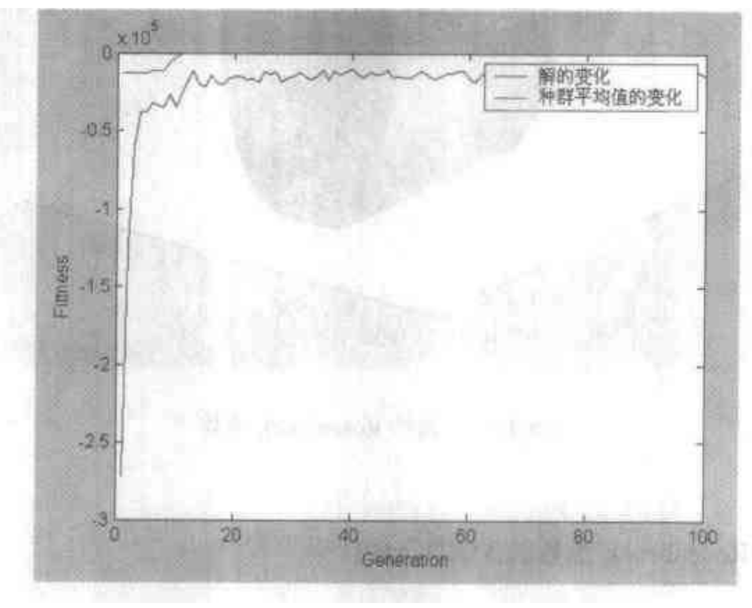



图 5-18　遗传算法寻优性能的跟踪图

2．Rosenbrock 函数的极小化问题

【问题】

Rosenbrock 函数是一个典型的优化问题，表达式如（式 5.4）所示。

$$f(x)=\sum_{i=1}^{n-1}(100*(x_{i+1}-x_i^2)^2+(1-x_i)^2) \quad \text{（式 5.4）}$$

求解： $\min f(x) \qquad -2048 \leqslant x_i \leqslant 2048$

二维的 Rosenbrock 函数图形如图 5-19 所示。

例程 5-24 是绘制图 5-19 的 MATLAB 代码。

**例程 5-24**

```
[x1,x2] = meshgrid(-512:1:512);
f=100*(x2-x1.^2).^2+(1-x1).^2;
mesh(x1,x2,f);
xlabel('x1');
ylabel('x2');
zlabel('f(x1,x2)');
```

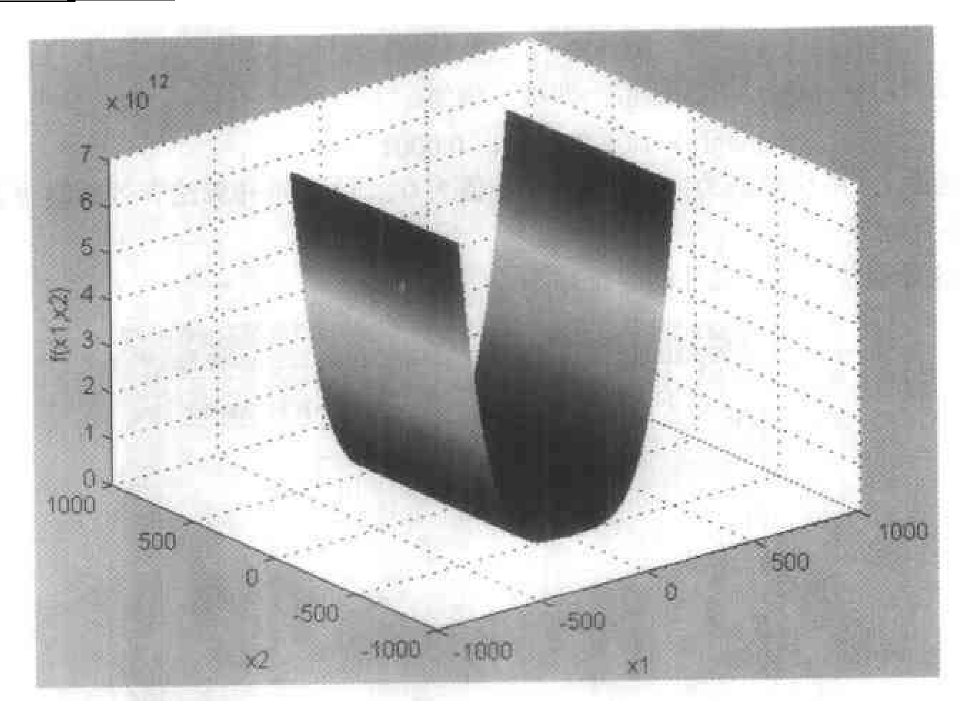



图 5-19 二维的 Rosenbrock 函数

**【程序清单】**

例程 5-25 是 Rosenbrock 函数的 MATLAB 代码。

**例程 5-25**

```
function [eval]=rosenbrock(sol)
numv = size(sol,2);
x=sol(1:numv);
summ=0;
```

```
for i=1:numv-1
    summ=summ+100*(x(i+1)-x(i).^2).^2+(1-x(i)).^2;
end
eval=summ;
```

例程 5-26 是计算 Rosenbrock 函数适值的 MATLAB 代码。

**例程 5-26**

```
function [sol,eval]=rosenbrockmin(sol,options)
numv = size(sol,2)-1;
x=sol(1:numv);
eval=rosenbrock(x);
eval=-eval;
```

例程 5-27 是遗传算法求解的 MATLAB 代码。

**例程 5-27**

```
%维数 n=4
%设置参数边界
bounds = ones(4,1)*[-512 512];

%遗传算法优化
[p,endPop,bestSols,trace]=ga(bounds,'rosenbrockMin');

%性能跟踪
plot(trace(:,1),trace(:,3),'b-')
hold on
plot(trace(:,1),trace(:,2),'r-')
xlabel('Generation');
ylabel('Fittness');
legend('解的变化','种群平均值的变化');
```

**【结果输出】**

（1）最优解为：p =

0.9836
0.9675
0.9360
0.8761

（2）此时 Rosenbrock 函数的适值，即极小值为：

Rosenbrock(p)=-0.0054

而在理论上，最优解应为：p=1 1 1，极小值为 0。显然，遗传算法有效地解决了 Rosenbrock 函数的极小化问题。

图 5-20 是遗传算法寻优性能的跟踪图。

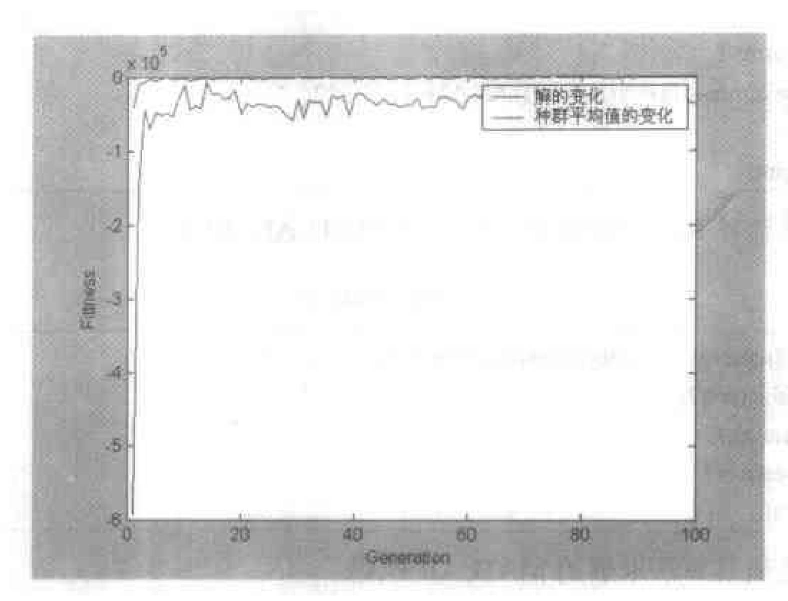



图 5-20　遗传算法寻优性能的跟踪图

### 3. Griewangk 函数的极小化问题

【问题】

Griewangk 函数表达式如（式 5.5）所示。

$$f(x)=\sum_{i=1}^{n}\frac{x_i^2}{4000}-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1 \qquad \text{（式 5.5）}$$

求解： $\min f(x)$　　　$-600 \leqslant x_i \leqslant 600$

二维的 Griewangk 函数图形如图 5-21 所示。

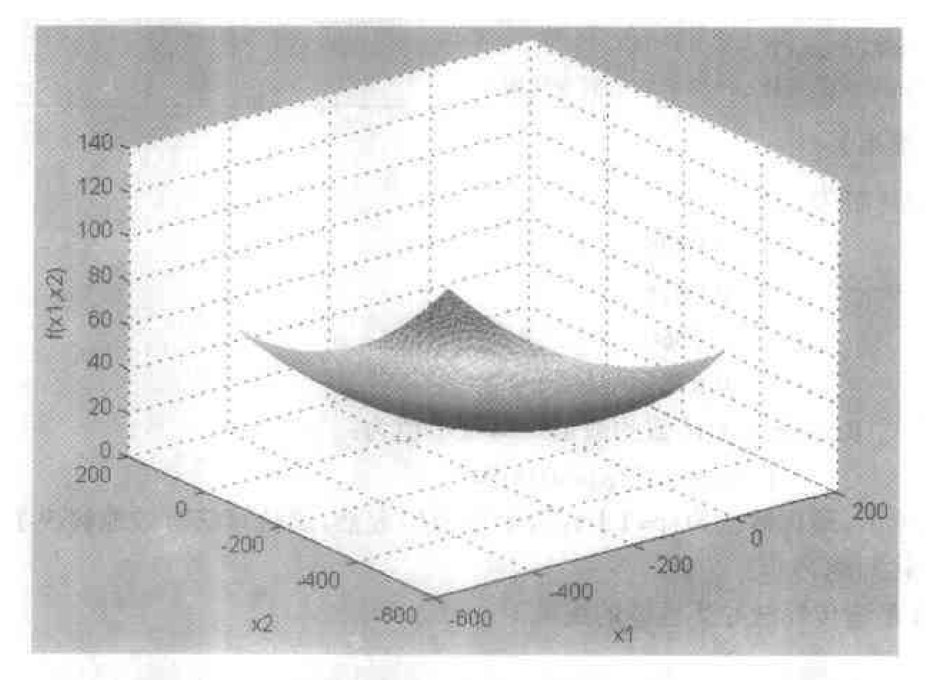



图 5-21　二维的 Griewangk 函数

例程 5-28 是绘制图 5-21 的 MATLAB 代码。

例程 5-28

```
[x1,x2] = meshgrid(-500:5:50);
f=x1.^2/4000+x2.^2/4000-cos(x1)*cos(x2/sqrt(2))+1;
mesh(x1,x2,f);
xlabel('x1');
ylabel('x2');
zlabel('f(x1,x2)');
```

【程序清单】

例程 5-29 是 Griewangk 函数的 MATLAB 代码。

例程 5-29

```
function [eval]=griewangk(sol)
numv = size(sol,2);
x=sol(1:numv);
multi=1;
for i=1:numv
   multi=multi*cos(x(i)/sqrt(i));
end
eval=sum(x.^2/4000)-multi+1;
```

例程 5-30 是计算 Griewangk 函数适值的 MATLAB 代码。

例程 5-30

```
function [sol,eval]=griewangkmin(sol,options)
numv = size(sol,2)-1;
x=sol(1:numv);
eval=griewangk(x);
eval=-eval;
```

例程 5-31 是遗传算法求解的 MATLAB 代码。

例程 5-31

```
%维数 n=6
%设置参数边界
bounds = ones(6,1)*[-512 512];

%遗传算法优化
[p,endPop,bestSols,trace]=ga(bounds,'griewangkmin');

%性能跟踪
plot(trace(:,1),trace(:,3),'b-')
hold on
plot(trace(:,1),trace(:,2),'r-')
```

```
xlabel('Generation');
ylabel('Fittness');
legend('解的变化','种群平均值的变化');
```

【结果输出】

图 5-22 是遗传算法寻优性能的跟踪图。

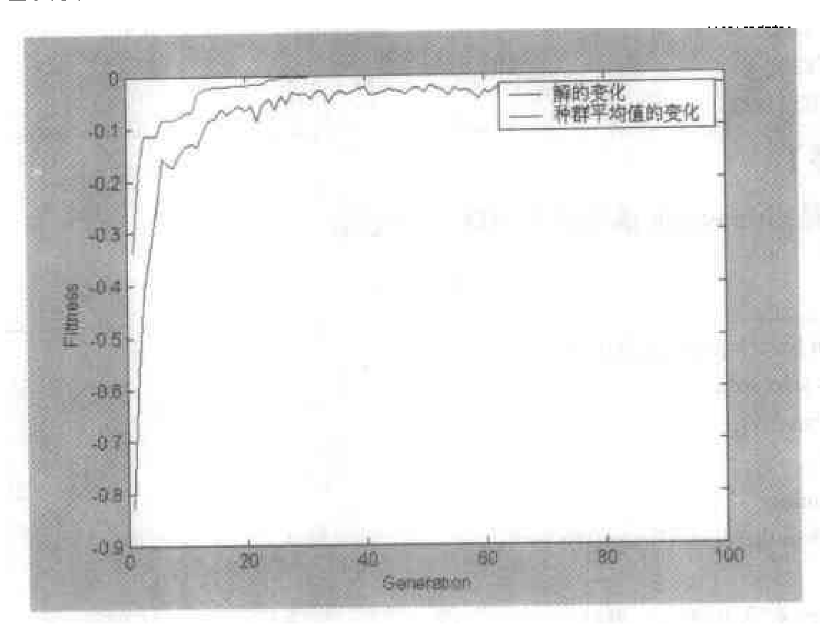



图 5-22 遗传算法寻优性能的跟踪图

（1）最优解为：p =

0.0002
-0.0004
-0.0003
0.0006
-0.0022
0.0002

（2）此时 Griewangk 函数的适值，即极小值为：

Griewangk(p)=-0.0000

而在理论上，最优解应为：p=0 0 0，极小值为 0。显然，遗传算法有效地解决了 Griewangk 函数的极小化问题。

## 5.5.6 遗传算法在动态系统最优控制中的应用

动态系统控制问题很复杂，也很难解决。使用不同的具体动态优化方法（比如 Hamiltonian）常常是繁琐的，并且或多或少地存在一些缺陷。实现这些优化方法可能需要大量的数学计算手段的支持，甚至对系统的形式和规模大小也有一定的限制，只有那些合适的系统才能用优化方法来求解。本小节通过实例说明了遗传算法在动态系统优化控制中的应用。



1．双积分器问题

双积分器问题用（式 5.6）的状态方程描述如下：

$$\begin{cases} \dot{x}_1 = u \\ \dot{x}_2 = x_1 \\ y = x_2 \end{cases} \qquad \text{（式 5.6）}$$

时间区间：$0 \leqslant t \leqslant 1$

初始条件：$x_1(0)=0, x_2(0)=-1$

终止条件：$x_1(0)=0, x_2(0)=0$

目标函数：

$$\min f(u) = \int_{t=0}^{t=1} u(t)^2 \, dt$$

【分析】

双积分器问题可以用一个 simulink 模型或者用一个 s-函数来表示，目标函数则用这个函数中的参数来表达。其 simulink 模型表示如图 5-23 所示。

双积分器模型

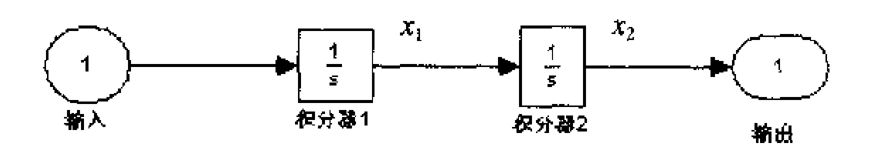



图 5-23　双积分器的 simulink 模型

例程 5-32 是图 5-23 的 MATLAB 代码。

例程 5-32

```
function [ret,x0,str]=simdopi1(t,x,u,flag);
sys = mfilename;
new_system(sys)
simver(1.2)
if(0 == (nargin + nargout))
      set_param(sys,'Location',[100,100,600,400])
      open_system(sys)
end;
set_param(sys,'algorithm',     'RK-45')
set_param(sys,'Start time',     '0.0')
```

```
set_param(sys,'Stop time',      '1')
set_param(sys,'Min step size', '0.001')
set_param(sys,'Max step size', '0.01')
set_param(sys,'Relative error','1e-3')
set_param(sys,'Return vars',     '')

add_block('built-in/Inport',[sys,'/','Inport'])
set_param([sys,'/','Inport'],...
          'position',[65,95,85,115])
add_block('built-in/Note',[sys,'/','Doppelintegrator'])
set_param([sys,'/','Doppelintegrator'],...
          'position',[225,10,230,15])
add_block('built-in/Note',[sys,'/','Steuerung'])
set_param([sys,'/','Steuerung'],...
          'position',[75,65,80,70])
add_block('built-in/Integrator',[sys,'/','Integrator1'])
set_param([sys,'/','Integrator1'],...
          'position',[175,95,195,115])
add_block('built-in/Integrator',[sys,'/','Integrator2'])
set_param([sys,'/','Integrator2'],...
          'Initial','-1',...
          'position',[280,95,300,115])
add_line(sys,[90,105;165,105])
add_line(sys,[200,105;270,105])

if (nargin | nargout)
    % 这里必须使用 feval 访问系统内存
    if (nargin > 3)
        if (flag == 0)
            eval(['[ret,x0,str]=',sys,'(t,x,u,flag);'])
        else
            eval(['ret =', sys,'(t,x,u,flag);'])
        end
    else
        [ret,x0,str] = feval(sys);
    end
end
```

**【程序清单】**

例程 5-33 是初始化种群的 MATLAB 代码。

例程 5-33

```
function [Chrom, Lind, BaseV] = crtbp(Nind, Lind, Base)
% Nind-种群的数目大小
% Lind-染色体的长度
% Base-染色体基
```



```
% Chrom-初始染色体

nargs = nargin ;
if nargs >= 1, [mN, nN] = size(Nind) ; end
if nargs >= 2, [mL, nL] = size(Lind) ; end
if nargs == 3, [mB, nB] = size(Base) ; end

if nN == 2
    if (nargs == 1)
        Lind = Nind(2) ; Nind = Nind(1) ; BaseV = crtbase(Lind) ;
    elseif (nargs == 2 & nL == 1)
        BaseV = crtbase(Nind(2),Lind) ; Lind = Nind(2) ; Nind = Nind(1) ;
    elseif (nargs == 2 & nL > 1)
        if Lind ~= length(Lind), error('Lind and Base disagree'); end
        BaseV = Lind ; Lind = Nind(2) ; Nind = Nind(1) ;
    end
elseif nN == 1
    if nargs == 2
        if nL == 1, BaseV = crtbase(Lind) ;
        else, BaseV = Lind ; Lind = nL ; end
    elseif nargs == 3
        if nB == 1, BaseV = crtbase(Lind,Base) ;
        elseif nB ~= Lind, error('Lind and Base disagree') ;
        else BaseV = Base ; end
    end
else
    error('Input parameters inconsistent') ;
end

Chrom = floor(rand(Nind,Lind).*BaseV(ones(Nind,1),:)) ;
```

例程5-34是进行染色体选择的MATLAB代码。

例程5-34

```
function NewChrIx = sus(FitnV,Nsel);
% FitnV-种群中每个染色体的适值
% Nsel-选择的染色体数目

[Nind,ans] = size(FitnV);
%  进行随机采样
cumfit = cumsum(FitnV);
trials = cumfit(Nind) / Nsel * (rand + (0:Nsel-1)');
Mf = cumfit(:, ones(1, Nsel));
Mt = trials(:, ones(1, Nind))';
[NewChrIx, ans] = find(Mt < Mf & [ zeros(1, Nsel); Mf(1:Nind-1, :) ] <= Mt);

%得到新的种群
```

```
[ans, shuf] = sort(rand(Nsel, 1));
NewChrIx = NewChrIx(shuf);
```

例程 5-35 是进行染色体交叉的 MATLAB 代码。

例程 5-35

```
function NewChrom = xovsp(OldChrom, XOVR);
% OldChrom-父代种群的染色体
% XOVR-交叉率

if nargin < 2, XOVR = NaN; end
% 得到新的染色体
NewChrom = xovmp(OldChrom, XOVR, 1, 0);
```

例程 5-36 进行染色体变异的 MATLAB 代码。

例程 5-36

```
function NewChrom = mut(OldChrom,Pm,BaseV)
% OldChrom-当代种群的染色体
% Pm-变异概率
% BaseV-染色体基

[Nind, Lind] = size(OldChrom) ;
if nargin < 2, Pm = 0.7/Lind ; end
if isnan(Pm), Pm = 0.7/Lind; end
if (nargin < 3), BaseV = crtbase(Lind);   end
if (isnan(BaseV)), BaseV = crtbase(Lind);   end
if (isempty(BaseV)), BaseV = crtbase(Lind);   end
if (nargin == 3) & (Lind ~= length(BaseV))
    error('OldChrom and BaseV are incompatible'), end
% 生成变异屏蔽矩阵
BaseM = BaseV(ones(Nind,1),:) ;

% 进行变异操作
NewChrom = rem(OldChrom+(rand(Nind,Lind)<Pm).*ceil(rand(Nind,Lind).*(BaseM-1)),BaseM);
```

例程 5-37 是定义目标函数的 MATLAB 代码。

例程 5-37

```
function [ObjVal,t,x] = objdopi(Chrom,sswitch);
% Chrom-当代种群的染色体
% sswitch-选择变量

% 目标函数维数
Dim = 20;
TSTART = 0;
TEND = 1;
```

```
STEPSIMU = min(0.1,abs((TEND-TSTART)/(Dim-1)));
TIMEVEC = linspace(TSTART,TEND,Dim)';

%初始条件
XINIT = [ 0; -1];
% 终止条件
XEND = [ 0; 0];

%控制权值
XENDWEIGHT = 12 * [1; 1];
UWEIGHT = [0.5];

% 计算种群参数
 [Nind,Nvar] = size(Chrom);
 if Nind == 0
        if sswitch == 2
            ObjVal = ['Double Integrator (sim)-' int2str(Dim)];
         elseif sswitch == 3
            ObjVal = 2;
        else
            % 变量的边界
            ObjVal1 = [-15; 15];
            ObjVal = rep(ObjVal1,[1 Dim]);
end
    elseif   Nvar == Dim
        ObjVal = zeros(Nind,1);
        for indrun = 1:Nind
            steuerung = [TIMEVEC [Chrom(indrun,:)]'];
            [t x] = rk23('simdopi1',[TSTART TEND],[],[1e-3;STEPSIMU;STEPSIMU],steuerung);
            ObjVal(indrun) = sum(XENDWEIGHT .* abs( x(size(x,1),:)' - XEND )) + ...
                              (UWEIGHT / (Dim-1) * trapz(Chrom(indrun,:).^2));
        end
    else
        error('size of matrix Chrom is not correct for function evaluation');
    end
```

例程 5-38 是求解的 MATLAB 代码。

**例程 5-38**

```
%每个种群中的个体数目
NIND = 20;
%最大代数
MAXGEN = 100;
GGAP = .8;

SEL_F = 'sus';        % 选择函数
XOV_F = 'xovsp';      % 交叉函数
```

```
MUT_F = 'mut';        % 变异函数
OBJ_F = 'objfun1';    % 目标函数

% 目标函数的边界
   FieldDR = feval(OBJ_F,[],1);

% 目标函数变量的数目
   NVAR = size(FieldDR,2);

% 描述矩阵
   PRECI = 20;
   FieldDD = [rep([PRECI],[1, NVAR]);...
                 FieldDR;...
                 rep([1; 0; 1 ;1], [1, NVAR])];

% 生成种群
   Chrom = crtbp(NIND, NVAR*PRECI);
gen = 0;
Best = NaN*ones(MAXGEN,1);
% 遗传迭代
while gen < MAXGEN,
% 计算目标函数
      ObjV = feval(OBJ_F,bs2rv(Chrom, FieldDD));
      Best(gen+1) = min(ObjV);
      plot(log10(Best),'bo');
      xlabel('generation');
      ylabel('log10(Best)');

   % 适度分配
      FitnV = ranking(ObjV);
   % 选择个体
      SelCh = select(SEL_F, Chrom, FitnV, GGAP);
   % 交叉
      SelCh=recombin(XOV_F, SelCh);
   % 变异
      SelCh=mutate(MUT_F, SelCh);
   %插入子代
      Chrom = reins(Chrom, SelCh);
      gen=gen+1;
   end
```

**【结果输出】**

图 5-24 表示的是遗传算法求解的迭代过程。

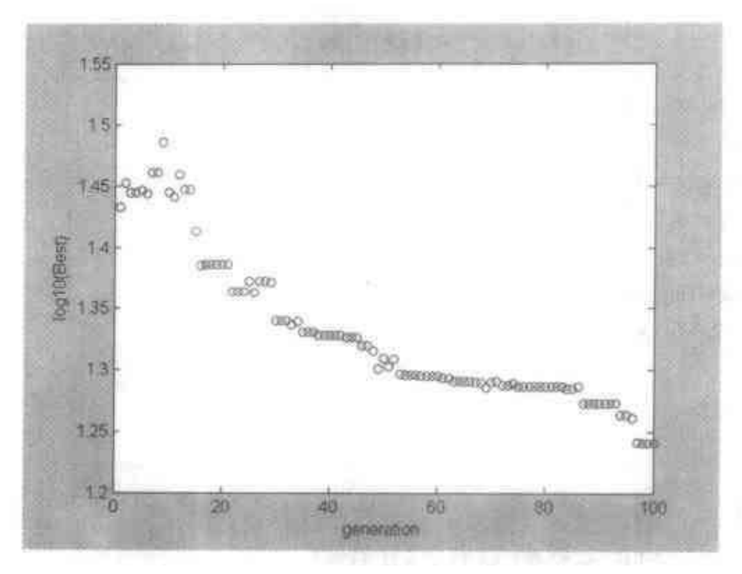



图 5-24 遗传算法求解的迭代过程

**2. 二次线性系统最优控制**

假设二阶线性系统是一维的，其表达式如下：

$$x(k+1)=a*x(k)+b*u(k) \qquad k=1,2,\cdots,N$$

目标函数定义为：

$$f(x,u)=q*x(N+1)^2+\sum_{k=1}^{N}(s*x(k)^2+r*u(k)^2)$$

参数设置如表 5-8 所示。

表 5-8 参数设置

| $N$ | $x(0)$ | $s$ | $r$ | $q$ | $a$ | $b$ |
|---|---|---|---|---|---|---|
| 45 | 100 | 1 | 1 | 1 | 1 | 1 |

**【分析】**

此二阶线性系统相当于带有一个正反馈的单积分器，其连续形式可以用一个 simulink 模型来表示，如图 5-25 所示。

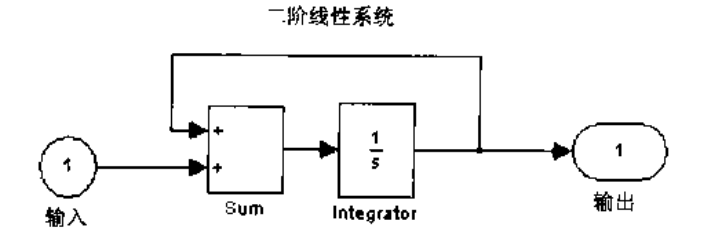



图 5-25 二阶线性系统的 simulink 模型

例程 5-39 是其 S-函数的.m 文件。

例程 5-39

```
function [sys, x0] = simdopi2(t, x, u, flag);
%系统描述
    if abs(flag) == 1
      sys(1) = u(1);
      sys(2) = x(1);
    elseif abs(flag) == 0
      sys=[2,0,0,1,0,0]; x0 = [0; -1];
    else
      sys = [];
    end
```

【程序清单】

例程 5-40 是定义目标函数的 MATLAB 代码。

例程 5-40

```
function ObjVal = objlinq(Chrom,sswitch);

% 目标函数的维数
    Dim = 45;

%参数值
    x0 = 100;                    % 初始点
    var = 1;
    Para = [   1     1     1     1     1     16180.3399];
    s = Para(var,1); r = Para(var,2); q = Para(var,3);
    a = Para(var,4); b = Para(var,5); GlobalMinimum = Para(var,6);

% 染色体大小
    [Nind,Nvar] = size(Chrom);

% 检验染色体大小
    % 如果染色体为空，则定义边界矩阵大小和值
    if Nind == 0
        %返回输出图形的标题文字
        if sswitch == 2
            ObjVal = ['Linear-quadratic problem (dis)-' int2str(Dim)];
        %返回全局最小值
        elseif sswitch == 3
            ObjVal = GlobalMinimum;
        % 定义边界矩阵大小和值
        else
            %上界和下界
            ObjVal1 = [-100 -70 -50; 20 20 20];
            ObjVal = [ObjVal1 rep([-30;20],[1 Dim-3])];
```

# 附录 C　MATLAB 6.5 安装问题指南

## C.1　MATLAB 6.5 为什么安装后不能启动

经常有朋友遇到这个问题，其实在 Windows 98 或其第 2 版下安装 MATLAB 6.5 应该没有问题（也有一部分可能会在安装了 IE 5.5 或更高版本并更新了 Windows 的一些字库以后出现），而在 Windows Me 或 Windows 2000 下安装 MATLAB 6.5 不能启动的主要原因是它的部分中文字库和操作系统的字库重名造成的，下面给出几种可行的解决方法以供选择。

**1．更改区域设置**

在控制面板里有“区域设置”一项，一般的中文 Windows 操作系统上默认的区域设置是“**中文（中国）**”，如图 C-1 所示。这里我们只要把它更改为“**英语（美国）**”之后重新启动计算机，MATLAB 6.5 就可以正常运行了。不用担心，改过区域设置之后，操作系统仍然是中文操作系统，并不会给我们的使用带来太多的麻烦（当然有些小的显示等方面的问题）。这也是最省事的一种解决方法了。




图 C-1　更改区域设置

**2．消除重名字库**

可以直接在操作系统中将重名字库删除，或是在 MATLAB 6.5 的字体配置文件中改变重名字库的相关信息。前一种删除方法将导致该字库从操作系统中被完全删掉，其他应用

```
    FigTitle = [feval(OBJ_F,[],2) '   (' int2str(SUBPOP) ':' int2str(MAXGEN) ') '];

% 清除 Bes 和存储矩阵内容
    %初始化存储最优结果的矩阵
        Best = NaN * ones(MAXGEN,3);
        Best(:,3) = zeros(size(Best,1),1);
    % 存储最优个体的矩阵
        IndAll = [];

%生成实值种群
Chrom = crtrp(SUBPOP*NIND,FieldDR);
% 计数变量归零
gen = 0;
termopt = 0;

% 计算给定种群的目标函数值
    ObjV = feval(OBJ_F,Chrom);
    %记录目标函数估计的次数
    Best(gen+1,3) = Best(gen+1,3) + NIND;

%迭代运算直至符合终止条件或达到了最大代数
    while ((gen < MAXGEN) & (termopt == 0)),

    % 保存最优和平均的目标函数值以及最优的个体
        [Best(gen+1,1),ix] = min(ObjV);
        Best(gen+1,2) = mean(ObjV);
        IndAll = [IndAll; Chrom(ix,:)];

    % 适应度分配
        FitnV = ranking(ObjV,[2 0],SUBPOP);

    % 从种群中选择个体
        SelCh = select(SEL_F, Chrom, FitnV, GGAP, SUBPOP);

    % 重新组合选择的个体
        SelCh=recombin(XOV_F, SelCh, XOVR, SUBPOP);

    % 变异
        SelCh=mutate(MUT_F, SelCh, FieldDR, [MUTR], SUBPOP);

    % 计算对应子代的目标函数值
        ObjVOff = feval(OBJ_F,SelCh);
        Best(gen+1,3) = Best(gen+1,3) + size(SelCh,1);

    %在种群中插入最优的子代以替换最劣的父代
        [Chrom, ObjV] = reins(Chrom, SelCh, SUBPOP, [1 INSR], ObjV, ObjVOff);
```

```
        gen=gen+1;

    % 检验. 如果最优的目标函数值接近于终止值,
    % 计算两者的差
        ActualMin = abs(min(ObjV) - GlobalMin);
    %如果最优的目标函数值小于终止值,
        if ((ActualMin < (TERMEXACT * abs(GlobalMin))) | (ActualMin < TERMEXACT))
            termopt = 1;
end
% 子代迁移
if ((termopt ~= 1) & (rem(gen,MIGGEN) == 0))
[Chrom, ObjV] = migrate(Chrom, SUBPOP, [MIGR, 1, 0], ObjV);
end
end
% 结果
    % 叠加目标函数估计的次数
    Results = cumsum(Best(1:gen,3));
    % 函数估计的次数、均值和最优结果
    Results = [Results Best(1:gen,2) Best(1:gen,1)];

% 绘图
    figure('Name',['Results of ' FigTitle]);
    subplot(2,1,1), plot(Results(:,1),Results(:,2),'-',Results(:,1),Results(:,3),':');
    subplot(2,1,2), plot(IndAll(gen-4:gen,:)');
    xlabel('u(k)');
    ylabel('f(x,u)');
```

**【结果输出】**

控制向量 $\boldsymbol{u}$ 与目标函数之间的关系如图 5-26 所示。

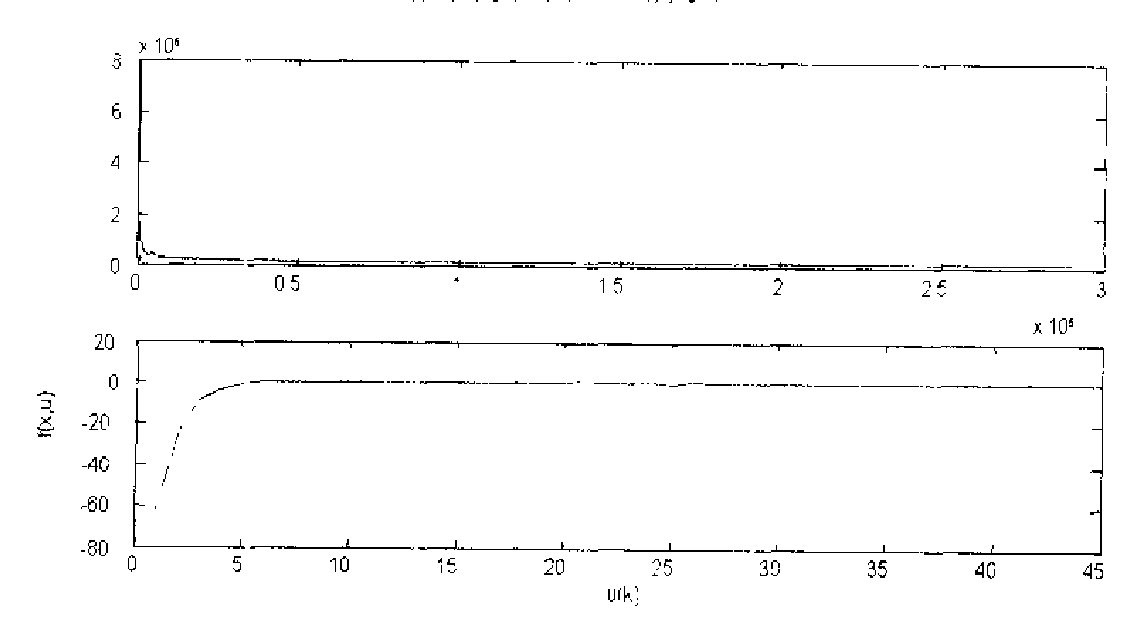



图 5-26　二阶线性系统的最优控制向量

3．Harvest 系统最优控制

Harvest 系统是一个一阶的离散方程，表达式如下：

$$x(k+1) = a * x(k) - u(k), \quad k = 1, \cdots, N$$
$$s.t. \quad x(0) = x(N)$$

目标函数：

$$\min f(u) = -\sum_{k=1}^{N} \sqrt{u(k)}$$

**【分析】**

理论的最优解是：$\min = -\sqrt{\dfrac{x(0)(a^N - 1)^2}{a^{N-1}(a-1)}}$

**【程序清单】**

例程 5-42 是目标函数的 MATLAB 代码。

例程 5-42

```
function ObjVal = objharv(Chrom,sswitch);
% 目标函数的维数
    Dim = 20;
% 参数设置
    a = 1.1;
    x0 = 100;
    xend = x0;
    XENDWEIGHT = 0.4/(Dim^0.6);

% 染色体的大小
    [Nind,Nvar] = size(Chrom);

% 染色体检查
    if Nind == 0
        %返回输出图形的标题文字
        if sswitch == 2
            ObjVal = ['HARVEST PROBLEM-' int2str(Dim)];
        %返回全局最小值
        elseif sswitch == 3
            ObjVal = -sqrt(x0*(a^Dim-1)^2/(a^(Dim-1)*(a-1)));
        %定义边界矩阵大小和值
        else
            % 变量的边界
            ObjVal1 = [0; 10*Dim];
            ObjVal = rep(ObjVal1,[1 Dim]);
```

```
        end
    elseif Nvar == Dim
        ObjVal = zeros(Nind,1);
        X = rep(x0,[Nind 1]);
        for irun = 1:Nvar,
            X = a*X - Chrom(:,irun);
        End
X;
ObjVal = -(sum(sqrt(Chrom)')' - XENDWEIGHT * abs(X-x0));
%否则，染色体格式错误
else
error('size of matrix Chrom is not correct for function evaluation');
    end
```

例程 5-43 是求解的 MATLAB 代码。

**例程 5-43**

```
GGAP = .8;            %代间隔，即每代产生多少个新个体
INSR = .9;            %插入率，即插入多少个子代
XOVR =   1;           %交叉率
SP = 2;               %选择压力
MUTR = 1;             %变异率
MIGR = 0.2;           %子代之间的迁移率
MIGGEN = 20;          %用于迁移的子代数目

TERMEXACT = 1e-4;     %当达到最小值时的终止值

SEL_F = 'sus';        %选择函数名
XOV_F = 'recdis';     %个体组合函数名
MUT_F = 'mutbga';     %变异函数名

OBJ_F = 'objharv';    %目标函数名

%得到目标函数的边界
   FieldDR = feval(OBJ_F,[],1);

%根据在目标函数中定义的变量的数目计算 SUBPOP 和 NIND
   NVAR = size(FieldDR,2);              %变量的数目
   SUBPOP = 2 * floor(sqrt(NVAR));      %子代的数目
   NIND = 20 + 5 * floor(NVAR/50);      %每个子代中个体的数目
   MAXGEN = 300 * floor(sqrt(NVAR)); %最大代数
   MUTR = MUTR / NVAR;                  %依赖于 NVAR 的变异率

%得到目标函数的最小值
   GlobalMin = feval(OBJ_F,[],3);

%目标函数图形输出的标题
```

```
    FigTitle = [feval(OBJ_F,[],2) '    (' int2str(SUBPOP) ':' int2str(MAXGEN) ') '];

%清除 Best 和存储矩阵内容
%初始化存储最优结果的矩阵
Best = NaN * ones(MAXGEN,3);
Best(:,3) = zeros(size(Best,1),1);
%存储最优个体的矩阵
IndAll = [];

%生成实值初始种群
    Chrom = crtrp(SUBPOP*NIND,FieldDR);

%计数变量归零
    gen = 0;
    termopt = 0;

%计算给定种群的目标函数值
    ObjV = feval(OBJ_F,Chrom);
    % 记录目标函数估计的次数
    Best(gen+1,3) = Best(gen+1,3) + NIND;

%迭代运算直至符合终止条件或达到了最大代数
    while ((gen < MAXGEN) & (termopt == 0)),

    %保存最优和平均的目标函数值以及最优的个体
        [Best(gen+1,1),ix] = min(ObjV);
        Best(gen+1,2) = mean(ObjV);
        IndAll = [IndAll; Chrom(ix,:)];

    %适应度分配
        FitnV = ranking(ObjV,[2 0],SUBPOP);

    %从种群中选择个体
        SelCh = select(SEL_F, Chrom, FitnV, GGAP, SUBPOP);

    %重新组合选择的个体
        SelCh=recombin(XOV_F, SelCh, XOVR, SUBPOP);

    %变异
        SelCh=mutate(MUT_F, SelCh, FieldDR, [MUTR], SUBPOP);

    %计算对应子代的目标函数值
        ObjVOff = feval(OBJ_F,SelCh);
        Best(gen+1,3) = Best(gen+1,3) + size(SelCh,1);

    %在种群中插入最优的子代以替换最劣的父代
```

```
        [Chrom, ObjV] = reins(Chrom, SelCh, SUBPOP, [1 INSR], ObjV, ObjVOff);

        gen=gen+1;
    %检验，如果最优的目标函数值接近于终止值，

%计算两者的差
ActualMin = abs(min(ObjV) - GlobalMin);

%如果最优的目标函数值小于终止值，
if ((ActualMin < (TERMEXACT * abs(GlobalMin))) | (ActualMin < TERMEXACT))
            termopt = 1;
        end

    %子代迁移
        if ((termopt ~= 1) & (rem(gen,MIGGEN) == 0))
            [Chrom, ObjV] = migrate(Chrom, SUBPOP, [MIGR, 1, 0], ObjV);
        end
    end

%结果
    %叠加目标函数估计的次数
    Results = cumsum(Best(1:gen,3));
    %函数估计的次数、均值和最优结果
    Results = [Results Best(1:gen,2) Best(1:gen,1)];

%绘图
    plot(IndAll(gen-4:gen,:)');
    xlabel('u(k)');
    ylabel('f(u)');
```

**【结果输出】**

控制向量 $\boldsymbol{u}$ 与目标函数之间的关系如图 5-27 所示。

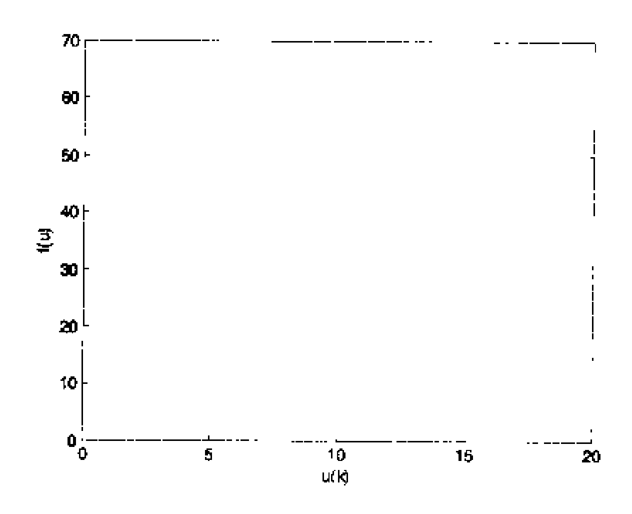



图 5-27 Harvest 系统的最优控制向量

4．Push-card 系统最优控制

Push-card 系统是一个二维的系统，表达式如下：

$$\begin{cases} x_1(k+1)=x_2(k) \qquad k=1,2,\cdots,N \\ x_2(k+1)=2*x_2(k)-x_1(k)+\dfrac{1}{N^2}u(k) \end{cases}$$

目标函数如下：

$$f(x,u)=-x_1(N+1)+\frac{1}{2*N}\sum_{k=1}^{N}u^2(k)$$

【分析】

理论的最优解是：$\min=-\dfrac{1}{3}+\dfrac{3*N-1}{6*N^2}+\dfrac{1}{2*N^3}\sum_{k=1}^{N-1}k^2$

【程序清单】

例程 5-44 是目标函数的 MATLAB 代码。

例程 5-44

```
function ObjVal = objpush(Chrom,sswitch);
% 目标函数的维数
  Dim = 20;
% 初始值
  x0 = [0 0];

% 计算染色体参数
  [Nind,Nvar] = size(Chrom);

% 染色体检查
  % 如果染色体为空，则定义边界矩阵的大小和值
  if Nind == 0
    %返回输出图形的标题文字
    if sswitch == 2
      ObjVal = ['PUSH-CART PROBLEM-' int2str(Dim)];
    %返回全局最小值
    elseif sswitch == 3
      ObjVal = -(1/3 - ((3*Dim-1)/(6*Dim^2)) - (1/(2*Dim^3))*sum((1:Dim-1).^2));
    %定义边界矩阵大小和值
    else
      %变量的上下边界
      ObjVal = [0; 5];
      ObjVal = rep(ObjVal,[1 Dim]);
    end
```



```
elseif Nvar == Dim
    ObjVal = zeros(Nind,1);
    X = rep(x0,[Nind 1]);
    for irun = 1:Nvar,
        Xsave = X;
        X(:,1) = Xsave(:,2);
        X(:,2) = 2 * X(:,2) - Xsave(:,1) + (1/Dim^2) * Chrom(:,irun);
    end
    X;
    ObjVal = -(X(:,1) - (1/(2*Dim)) * sum((Chrom.^2)')');
%否则，染色体格式错误
else
    error('size of matrix Chrom is not correct for function evaluation');
end
```

例程 5-45 是求解的 MATLAB 代码。

**例程 5-45**

```
GGAP = .8;            %代间隔，即每代产生多少个新个体
INSR = .9;            %插入率，即插入多少个子代
XOVR =   1;             %交叉率
SP = 2;               %选择压力
MUTR = 1;               %变异率
MIGR = 0.2;            %子代之间的迁移率
MIGGEN = 20;            %用于迁移的子代数目

TERMEXACT = 1e-4;     %当达到最小值时的终止值

SEL_F = 'sus';        %选择函数名
XOV_F = 'recdis';     %个体组合函数名
MUT_F = 'mutbga';      %变异函数名

OBJ_F = 'objpush';    %目标函数名

%得到目标函数的边界
   FieldDR = feval(OBJ_F,[],1);

%根据在目标函数中定义的变量的数目计算 SUBPOP 和 NIND
   NVAR = size(FieldDR,2);            %变量的数目
   SUBPOP = 2 * floor(sqrt(NVAR));    %子代的数目
   NIND = 20 + 5 * floor(NVAR/50);    %每个子代中个体的数目
   MAXGEN = 300 * floor(sqrt(NVAR)); %最大代数
   MUTR = MUTR / NVAR;                 %依赖于 NVAR 的变异率

%得到目标函数的最小值
GlobalMin = feval(OBJ_F,[],3);
```

```
    %目标函数图形输出的标题
FigTitle = [feval(OBJ_F,[],2) '   (' int2str(SUBPOP) ': int2str(MAXGEN) ') '];
%清除 Best 和存储矩阵内容
%初始化存储最优结果的矩阵
Best = NaN * ones(MAXGEN,3);
        Best(:,3) = zeros(size(Best,1),1);
    %存储最优个体的矩阵
        IndAll = [];
%生成实值初始种群
    Chrom = crtrp(SUBPOP*NIND,FieldDR);
%计数变量归零
    gen = 0;
    termopt = 0;
%计算给定种群的目标函数值
    ObjV = feval(OBJ_F,Chrom);
    %记录目标函数估计的次数
    Best(gen+1,3) = Best(gen+1,3) + NIND;
%迭代运算直至符合终止条件或达到了最大代数
    while ((gen < MAXGEN) & (termopt == 0)),
    %保存最优和平均的目标函数值以及最优的个体
        [Best(gen+1,1),ix] = min(ObjV);
        Best(gen+1,2) = mean(ObjV);
        IndAll = [IndAll; Chrom(ix,:)];
    %适应度分配
        FitnV = ranking(ObjV,[2 0],SUBPOP);

    %从种群中选择个体
        SelCh = select(SEL_F, Chrom, FitnV, GGAP, SUBPOP);

    %重新组合选择的个体
        SelCh=recombin(XOV_F, SelCh, XOVR, SUBPOP);
    %变异
        SelCh=mutate(MUT_F, SelCh, FieldDR, [MUTR], SUBPOP);

    %计算对应子代的目标函数值
        ObjVOff = feval(OBJ_F,SelCh);
        Best(gen+1,3) = Best(gen+1,3) + size(SelCh,1);

%在种群中插入最优的子代以替换最劣的父代
        [Chrom, ObjV] = reins(Chrom, SelCh, SUBPOP, [1 INSR], ObjV, ObjVOff);
        gen=gen+1;
%检验，如果最优的目标函数值接近于终止值，
%计算两者的差
ActualMin = abs(min(ObjV) - GlobalMin);
%如果最优的目标函数值小于终止值
if ((ActualMin < (TERMEXACT * abs(GlobalMin))) | (ActualMin < TERMEXACT))
```

```
            termopt = 1;
        end
    %子代迁移
        if ((termopt ~= 1) & (rem(gen,MIGGEN) == 0))
            [Chrom, ObjV] = migrate(Chrom, SUBPOP, [MIGR, 1, 0], ObjV);
        end
    end
%结果
    %叠加目标函数估计的次数
    Results = cumsum(Best(1:gen,3));
    %函数估计的次数、均值和最优结果
    Results = [Results Best(1:gen,2) Best(1:gen,1)];

%绘图
    plot(IndAll(gen-4:gen,:)');
    xlabel('u(k)');
    ylabel('f(u)');
```

**【结果输出】**

控制向量 $\boldsymbol{u}$ 与目标函数之间的关系如图 5-28 所示。

图 5-28　系统的最优控制向量



5.5.6 节例程中用到的其他函数请参见附录 D。



# 第6章　优化工具箱的工程应用实例

本章分别举例说明了 MATLAB 优化工具箱在生产计划规划、配料、投资决策、优化设计、信号处理、生物代谢优化，以及大规模优化等领域中的高级应用；向读者全面展示了应用 MATLAB 优化工具箱实现优化计算的全过程：问题的分析→建立数学模型→确定优化算法→运用优化工具箱函数→完成优化计算。

本章主要内容包括：

- 优化工具箱在生产计划规划中的应用
- 优化工具箱在配料中的应用
- 优化工具箱在资金投资领域中的应用
- 优化工具箱在优化设计中的应用
- 优化工具箱在信号处理中的应用
- 优化工具箱在生物代谢分析中的应用
- 优化工具箱在大规模规划中的应用

## 6.1　引　　言

从数学的角度看，最优化问题的求解首先应该建立数学模型，然后用相应的最优化方法进行求解。显然，如果数学模型建立得不好，则不管数学解法多么先进也难以求得符合实际情况的最优解。数学模型的建立应当比最优化的数学求解还重要。当量化地求解一个实际的最优化问题时，首先要把这个问题转化为一个数学问题，即建立数学模型；然后对建立的数学模型进行具体分析，选择合适的优化算法；最后根据选定的优化算法，编写计算程序进行求解。用 MATLAB 优化工具箱解决实际工程应用问题可概括为以下三个步骤。

（1）根据所提出的最优化问题，建立最优化问题的数学模型，确定变量，列出约束条件和目标函数（指标函数和性能函数）；

（2）对所建立的模型进行具体分析和研究，选择合适的最优化求解方法；

（3）根据最优化方法的算法，列出程序框图、选择优化函数和编写语言程序，用计算机求出最优解。

本章正是按照这三个步骤举例说明怎样应用 MATLAB 优化工具箱来解决不同领域的实际工程优化问题的。

## 6.2　优化工具箱在生产计划规划中的应用

在生产计划中，我们常常会遇到一些规划问题，借助 MATLAB，我们可以避免一些

繁琐的程序设计，从而很方便地解决这些问题。以下就给出一些例子，讲述在农业生产和工厂生产中所碰到的一些问题的解决办法。

### 6.2.1 农业生产计划的优化安排

某村计划在100公顷的土地上种植A、B、C三种农作物，可以提供的劳力、粪肥和化肥等资源的数量，种植每公顷作物所需这三种资源的数量，以及能够获得的利润如表6-1所示。

表6-1 种植投入产出

| | 用　工 | 粪肥（吨） | 化肥（千克） | 利润（元） |
|---|---|---|---|---|
| A | 450 | 35 | 350 | 1500 |
| B | 600 | 25 | 400 | 1200 |
| C | 900 | 30 | 300 | 1800 |
| 可提供资源 | 63000 | 3300 | 33000 | |

其中一个劳动力干一天为1个工。现在要求为该村制定一个农作物的种植计划，确定每种农作物的种植面积，使得总利润最大。

1．问题的提出

在安排农业生产计划时，一个重要的约束条件就是可提供资源的有限性，它是考虑问题的基点。要使总利润最大，一方面希望多种植利润高的农作物；但另一方面，利润高的农作物所需的资源也多，从而减少了农作物的种植面积，这样可能会降低总利润。因此本问题的实质就是一个在资源受限的条件下，寻求最大总利润的约束优化问题，可用MATLAB优化工具箱进行求解

2．模型符号约定

$x_1$：农作物A的种植面积；

$x_2$：农作物B的种植面积；

$x_3$：农作物C的种植面积；

*INCOME*：总利润。

3．模型的建立

由题意，总利润为：

$$INCOME = 1500 * x_1 + 1200 * x_2 + 1800 * x_3$$

资源受限条件：

（1）土地限制：总的种植面积为100公顷，即

$$x_1 + x_2 + x_3 = 100$$

（2）劳力限制：种植三种农作物的用工之和不能超过允许值63 000个工，即

$$450 * x_1 + 600 * x_2 + 900 * x_3 \leqslant 63000$$

（3）粪肥限制：不超过 3 300 吨，即

$$35*x_1+25*x_2+30*x_3\leqslant 3300$$

（4）化肥限制：不超过 33 000 吨，即

$$350*x_1+400*x_2+300*x_3\leqslant 33000$$

（5）另外，$x_1$，$x_2$，$x_3$ 不能为负值，即：

$$x_1,x_2,x_3\geqslant 0$$

综上所述，我们的问题就是在（1）～（5）的约束条件下，求最优解 $x_1,x_2,x_3$ 使得 *INCOME* 最大。因此，问题的数学模型建立如下：

$$\max_x INCOME=\max_x(1500*x_1+1200*x_2+1800*x_3)$$

$$\text{约束为}\begin{cases}x_1+x_2+x_3=100\\450x_1+600x_2+900x_3\leqslant 63000\\35x_1+25x_2+30x_3\leqslant 3300\\350x_1+400x_2+300x_3\leqslant 33000\\x_1,x_2,x_3\geqslant 0\end{cases}$$

显然，这是一个约束线性规划问题。

**4．模型的求解**

**【计算方法】**

本问题可以采用 MATLAB 优化工具箱中约束优化函数来求解，这里采用 fmincon 函数。为了与 fmincon 函数用法的标准形式保持一致，将原问题转化为：

$$\min_x f(\boldsymbol{x})=\min_x(-1500*x_1-1200*x_2-1800*x_3)$$

$$\text{约束为：}\begin{cases}\boldsymbol{A}*\boldsymbol{x}\leqslant \boldsymbol{b}\\\boldsymbol{Aeq}*\boldsymbol{x}=\boldsymbol{beq}\\\boldsymbol{lb}\leqslant \boldsymbol{x}\leqslant \boldsymbol{ub}\end{cases}$$

其中：

$$\boldsymbol{A}=\begin{bmatrix}450&600&900\\35&25&30\\350&400&300\end{bmatrix},\quad \boldsymbol{b}=\begin{bmatrix}63000\\3300\\33000\end{bmatrix},\quad \boldsymbol{Aeq}=\begin{bmatrix}1&1&1\end{bmatrix},\quad \boldsymbol{beq}=100$$

$$\boldsymbol{x}=\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix},\quad \boldsymbol{lb}=\begin{bmatrix}0\\0\\0\end{bmatrix},\quad \boldsymbol{ub}=\begin{bmatrix}100\\100\\100\end{bmatrix}$$

**【程序清单】**

例程 6-1 是求解的 MATLAB 代码。

例程 6-1

```
function f=fun61(x)
f=-1500*x(1)-1200*x(2)-1800*x(3);

%不等式约束
A=[450 600 900;35 25 30;350 400 300];
b=[63000;3300;33000];
%等式约束
Aeq=[1 1 1];
beq=[100];
%边界约束
lb=[0;0;0];
ub=[100;100;100];
%标准算法
options=optimset('largescale','off');
%初始点
x0=[30 30 40];
%优化函数调用
[x,fval]=fmincon('fun61',x0,A,b,Aeq,beq,lb,ub,[],options);
```

**【结果输出】**

```
Optimization terminated successfully:
Search direction less than 2*options.TolX and
maximum constraint violation is less than options.TolCon
Active Constraints:
     1
     3
     8
x =
        60.0000        0      40.0000
fval =
       -162000
```

因此，最优的农业生产计划如表 6-2 所示。

表 6-2　最优农业生产计划

| | A | B | C |
|---|---|---|---|
| 种植面积 | 60 公顷 | 0 | 40 公顷 |
| 利润 | 90000 元 | 0 | 72000 元 |

总利润为：16 2000 元



### 6.2.2 工厂生产的优化调度

某水泥厂采用湿法工艺生产水泥（其工艺说明如表 6-3 所示），水泥厂一般自己开采石灰石矿及黏质砂岩矿，外购铁矿原料（这个水泥厂使用外购价格便宜的硫酸铁渣），石灰石矿中夹有浮土层可以利用 10%，在极特殊配料困难情况下，才外购少量白砂（硅石）来满足配料要求。

表 6-3 湿法生产水泥的工艺说明

| 石灰石 | 浮土 | 黏质砂岩 | 硫酸铁渣 |
| --- | --- | --- | --- |
| 78% | 8% | 11% | 3% |

利用以上原料混合配料加水分 35%，在原料磨机中将湿法粉碎成料浆送入多个不同成分（高、低、中）的料浆库（并用压缩空气法搅拌均匀，防止沉淀），第二次将合格料浆送入搅拌池，用料浆泵送入水泥旋转窑中煅烧成合格半成品熟料（clinker）。

水泥厂现使用三组磨机生产三种型号的水泥，其生产效率如表 6-4 和表 6-5 所示。

表 6-4 磨机的生产效率 （单位：万吨/天）

| | 625 号 | 525 号 | 425 号 |
| --- | --- | --- | --- |
| A 组磨 1 台 | 0.042 | 0.043 | 0.044 |
| B 组磨 2 台 | 0.035*2=0.07 | 0.036*2=0.073 | 0.037*2=0.074 |
| C 组磨 1 台 | 0.0365 | 0.0375 | 0.0385 |

表 6-5 不同型号水泥的配料和生产利润

| | 625 号 | 525 号 | 425 号 |
| --- | --- | --- | --- |
| 熟料 | 0.95 | 0.68 | 0.54 |
| 干矿渣 | 0 | 0.27 | 0.42 |
| 生产利润 | 28 万元/万吨 | 12 万元/万吨 | 6 万元/万吨 |

另外，该厂在制定生产规划时面临以下情况：

（1）确定年产量以 36 万 5 千吨来核算。

（2）625 号普通水泥生产 4 万吨以内。

（3）525 号矿渣水泥不定。

（4）425 号矿渣水泥按市场需要有 16 万、15 万、14 万等 3 种情况。

（5）A 组磨机每台工作天数≤310

B 组磨机每台工作天数≤310

C 组磨机每台工作天数≤150

（6）熟料供给量 24 万吨。

（7）年干矿渣供应量 10 万吨。

（8）预计年生产利润为 350 万元。

现要求为该厂制定一个年生产规划，以便能够充分利用资源并且实现年产量和年生产

利润。

1．问题分析

不同型号的水泥的生产利润不同，不同组别的磨机生产不同型号的水泥的效率也不同，不同型号的水泥所需的原材料也不同。因此，制定生产规划就是要确定：采用哪一组磨机生产什么型号的水泥；每一组磨机的生产天数是多少；每一型号水泥的年生产量是多少，使得在决策过程中受到一定的实际情况制约的情况下，能够充分利用给定的资源，实现预计年产量和年生产利润。因此，我们可以建立一个多目标规划模型，用 MATLAB 优化工具箱来解决。

2．模型符号约定

- $x_{11}$：表示 A 组磨机生产 C1 产品(625 号)的年运行天数/台；
- $x_{12}$：表示 A 组磨机生产 C2 产品(525 号)的年运行天数/台；
- $x_{13}$：表示 A 组磨机生产 C3 产品(425 号)的年运行天数/台；
- $x_{21}$：表示 B 组磨机生产 C1 产品(625 号)的年运行天数/台；
- $x_{22}$：表示 B 组磨机生产 C2 产品(525 号)的年运行天数/台；
- $x_{23}$：表示 B 组磨机生产 C3 产品(425 号)的年运行天数/台；
- $x_{31}$：表示 C 组磨机生产 C1 产品(625 号)的年运行天数/台；
- $x_{32}$：表示 C 组磨机生产 C2 产品(525 号)的年运行天数/台；
- $x_{33}$：表示 C 组磨机生产 C3 产品(425 号)的年运行天数/台；
- $W_1$：C1 产品产量；
- $W_2$：C2 产品产量；
- $W_3$：C3 产品产量；
- $W$：水泥年总产量；
- $W_4$：工矿渣供应量；
- $W_5$：熟料供应量；
- $T_1$：A 组磨机每台工作天数；
- $T_2$：B 组磨机每台工作天数；
- $T_3$：C 组磨机每台工作天数；
- TOTLE：年生产利润。

3．模型的建立

（1）问题

生产利润：

$$TOTLE= 28*(0.042x_{11}+0.07x_{21}+0.0365x_{31})+12*(0.043x_{12}+0.072x_{22}+0.0375x_{32})+6*(0.044x_{13}+0.074x_{23}+0.0385x_{33})$$

$$=1.176x_{11}+0.516x_{12}+0.264x_{13}+1.96x_{21}+0.864x_{22}+0.444x_{23}+1.008x_{31}+0.45x_{32}+0.231x_{33}$$

产量：

$W_1 = 0.042x_{11}+0.07x_{21}+0.0365x_{31}$ [illegible]4

$W_2 = 0.043x_{12} + 0.072x_{22} + 0.0375x_{32} \quad 14 \leqslant W_2 \leqslant 22$

$W_3 = 0.044x_{13} + 0.074x_{23} + 0.0385x_{33} \leqslant 16$

总产量：

$W = W_1 + W_2 + W_3$

$= 0.042x_{11} + 0.043x_{12} + 0.044x_{13} + 0.07x_{21} + 0.072x_{22} + 0.074x_{23} + 0.0365x_{31} + 0.0375x_{32} + 0.0385x_{33}$

干矿渣供应量：

$W_4 = 0.27W_2 + 0.42W_3$

$= 0.0116x_{12} + 0.0185x_{13} + 0.0194x_{22} + 0.0302x_{23} + 0.1013x_{32} + 0.0162x_{33}$

熟料供应量：

$W_5 = 0.95W_1 + 0.68W_2 + 0.54W_3$

$= 0.0399x_{11} + 0.02924x_{12} + 0.02376x_{13} + 0.0665x_{21} + 0.04896x_{22} + 0.03996x_{23} + 0.0342x_{31} + 0.0255x_{32} + 0.02079x_{33}$

工作时间限制：

$T_1 = x_{11} + x_{12} + x_{13} \leqslant 310$

$T_2 = x_{21} + x_{22} + x_{23} \leqslant 310$

$T_3 = x_{31} + x_{32} + x_{33} \leqslant 150$

（2）建立优化目标

第一个优化目标：必须充分利用干矿渣，也即

$$\min f_1(x) = \min(10 - W_4)$$

第二个优化目标：必须充分利用熟料，也即

$$\min f_2(x) = \min(24 - W_5)$$

第三个优化目标：必须尽量完成年产量，也即

$$\min f_3(x) = \min(36.5 - W)$$

第四个优化目标：必须尽量完成年生产利润，也即

$$\min f_4(x) = \min(350 - TOTLE)$$

（3）建立多目标规划模型

多目标向量函数：

$$F(x) = [f_1(x), f_2(x), f_3(x), f_4(x)]$$

目标值：　　goal=[0.001，0.001，0.001，0.001]

且满足约束：

$$0.042x_{11} + 0.07x_{21} + 0.0365x_{31} \leqslant 4$$

$$0.043x_{12} + 0.072x_{22} + 0.0375x_{32} \leqslant 22$$

$$-0.043\,x_{12}-0.072\,x_{22}-0.0375\,x_{32}\leqslant 14$$

$$0.044\,x_{13}+0.074\,x_{23}+0.0385\,x_{33}\leqslant 16$$

$$x_{11}+x_{12}+x_{13}\leqslant 310$$

$$x_{21}+x_{22}+x_{23}\leqslant 310$$

$$x_{31}+x_{32}+x_{33}\leqslant 150$$

4．模型的求解

**【计算方法】**

写成 MATLAB 中求解多目标规划问题的标准形式：

$$\min_{x,\gamma}\gamma,$$

$$\text{使得：}\begin{cases}F(x)-weight*\gamma\leqslant \boldsymbol{goal}\\ \boldsymbol{A}*\boldsymbol{x}\leqslant \boldsymbol{b}\\ 0\leqslant \boldsymbol{x}\end{cases}$$

其中：

$$\boldsymbol{x}=[x_{11},x_{12},x_{13},x_{21},x_{22},x_{23},x_{31},x_{32},x_{33}]'$$

**【程序清单】**

例程 6-2 是求解的 MATLAB 代码。

例程 6-2

```
function f=y(x)
f(1)=10-0.0116*x(2)-0.0185*x(3)-0.0194*x(5)-0.0302*x(6)- 0.1013*x(8)-0.0162*x(9);
f(2)=24-0.0399*x(1)-0.02924*x(2)-0.02376*x(3)-0.0665*x(4)-...
.04896*x(5)-0.03996*x(6)-0.0342*x(7)-0.0255*x(8)-0.02079*x(9);
f(3)=36.5-0.042*x(1)-0.043*x(2)-0.044*x(3)-0.07*x(4)-0.072*...x(5)-0.074*x(6)-0.0365*x(7)-
0.0375*x(8)-0.0385*x(9);
f(4)=350-1.176*x(1)-0.516*x(2)-0.264*x(3)-1.96*x(4)-0.864*...x(5)-0.444*x(6)-1.008*x(7)-
0.45*x(8)-
0.231*x(9);

A=[0.042,0,0,0.07,0,0,0.0365,0,0;
    0,0.043,0,0,0.072,0,0,0.0375,0;
    0,-0.043,0,0,-0.072,0,0,-0.0375,0;
    1,1,1,0,0,0,0,0,0;
    0,0,0,1,1,1,0,0,0;
    0,0,0,0,0,0,1,1,1];
b=[4,23,-14,16,310,310.150]';

goal=[0.001,0.001,0.001,0.001]';
```

```
weight=abs(goal);

lb=0;
x0=[10,10,10,10,10,10,10,10,10]';

x=fgoalattain('f',x0,goal,weight,A,b,[],[],lb,[]);
```

【结果输出】

```
x =
    71.4311
    218.0722
    0.0000
    0.0000
    175.2632
    134.7412
    0.0000
    0.0000
    0.0000
```

于是有：

A 组磨机生产 625 号水泥年运行 71.43 天/台

A 组磨机生产 525 号水泥年运行 218.07 天/台

A 组磨机生产 425 号水泥年运行 0 天/台

B 组磨机生产 625 号水泥年运行 0 天/台

B 组磨机生产 525 号水泥年运行 175.26 天/台

B 组磨机生产 425 号水泥年运行 134.74 天/台

C 组磨机生产 625 号水泥年运行 0 天/台

C 组磨机生产 525 号水泥年运行 0 天/台

C 组磨机生产 425 号水泥年运行 0 天/台

625 号水泥生产 $0.042x_{11}+0.07x_{21}+0.0365x_{31}=3$ 万吨

525 号水泥生产 $0.043x_{12}+0.072x_{22}+0.0375x_{32}=22$ 万吨

425 号水泥生产 $0.044x_{13}+0.074x_{23}+0.0385x_{33}=9.97$ 万吨

干矿渣使用了 10 万吨

熟料使用了 23.19 万吨。

## 6.3 优化工具箱在配料中的应用

本节将介绍一个优化工具箱在配料问题中应用的例子。

大家知道，棉纺厂的主要原料是棉花，一般要占成本的 70%左右。棉花的品种、等级不同，价格也不同。因此，若采用品种好、等级高、价格贵的棉花来织成一种质量要求一般的棉纱，势必提高成本。所谓配棉问题，就是要根据棉纱的质量指标，采用各种价格不同的棉花，按一定比例配制成纱，使其既达到质量指标，又使总成本最低。

棉纱的质量指标一般由棉结和品质指标来决定．这两项指标都可用数量形式来表示，一般来说，棉结粒数越少越好，品质指标越大越好　但在具体生产过程中，它受棉花品种的现有量、生产技术、设备条件等多方面的制约

现有一个年纺纱能力为 15 000 锭的小厂的某一种产品 32D 纯棉纱的棉花配比、质量指标及单价，如表 6-6 所示

表 6-6　棉花配比、质量指标及单价

| 原料品名 | 单价（元/t） | 混合比（%） | 棉结数 | 品质指标 | 混棉单价（元/t） |
|---|---|---|---|---|---|
| 国棉 131 | 8 400 | 25 | 60 | 3 800 | 2 100 |
| 国棉 229 | 7 500 | 35 | 65 | 3 500 | 2 625 |
| 国棉 327 | 6 700 | 40 | 80 | 2 500 | 2 680 |
| 平均合计 | | 100 | 70 | 3 175 | 7 405 |

有关部门对 32D 纯棉纱规定的质量指标为：棉结不多于 70 粒，品质指标不小于 2 900，要求为该厂确定此品种的配棉方案，使得混棉单价最小。

1．问题的提出

棉纱的质量指标由棉结和品质指标来决定，这两项指标都可用数量形式来表示，一般来说，棉结粒数越少越好，品质指标越大越好。但在具体生产过程中，它受棉花品种的现有量、生产技术、设备条件等多方面的制约，如果采用高等级的棉花品种和复杂的生产技术等，这就会大大增加该棉花品种的价格，不利于厂家的生产销售，因此常常通过配棉来解决这一问题。就要根据给定棉纱的质量指标，采用各种价格不同的棉花，按一定比例配制成纱，使其既达到质量指标，又使总成本最低。

2．问题分析

题中给出了两个质量指标：棉结不多于 70 粒及品质指标不小于 2 900，要求的目标就是配棉后的棉纱单价最小，这实际上就是确定配棉方案，使得在给定质量指标的前提下配棉后的棉纱单价最小，也即是一个线性约束优化问题，可以用 MATLAB 优化工具箱来解决它

3．模型符号约定

- $x1$：国棉 131 的棉花配比
- $x2$：国棉 229 的棉花配比
- $x3$：国棉 327 的棉花配比
- $COST$：混棉单价

4．模型的建立

模型的目标函数使混棉单价最小，用 $x1$，$x2$，$x3$ 表示：

$$\min COST = 8400 * x1 + 7500 * x2 + 6700 * x3$$

质量指标要求可作为约束条件，即：



$$\begin{cases} 60*x1+65*x2+80*x3\leqslant 70 \\ 3800*x1+3500*x2+2500*x3\geqslant 2900 \end{cases}$$

再由变量的实际意义，有约束条件：

$$x1+x2+x3=1$$
$$x1,x2,x3\geqslant 0$$

所以，32D 纯棉的配棉问题的数学模型为：

$$\min COST=8400*x1+7500*x2+6700*x3$$

且满足：

$$\begin{cases} 60*x1+65*x2+80*x3\leqslant 70 \\ 3800*x1+3500*x2+2500*x3\geqslant 2900 \\ x1+x2+x3=1 \\ x1,x2,x3\geqslant 0 \end{cases}$$

**5．模型的求解**

**【计算方法】**

我们将对上述模型采用线性规划方法来求解，其格式转化为：

$$\min COST=\boldsymbol{f}^T*\boldsymbol{x}$$

且满足：

$$\begin{cases} \boldsymbol{A}*\boldsymbol{x}\leqslant \boldsymbol{b} \\ \boldsymbol{lb}\leqslant \boldsymbol{x}\leqslant \boldsymbol{ub} \\ \boldsymbol{Aeq}*\boldsymbol{x}=\boldsymbol{beq} \end{cases}$$

其中：

$$\boldsymbol{f}=\begin{bmatrix} 8400 \\ 7500 \\ 6700 \end{bmatrix},\quad \boldsymbol{x}=\begin{bmatrix} x1 \\ x2 \\ x3 \end{bmatrix},\quad \boldsymbol{A}=\begin{bmatrix} 60 & 65 & 80 \\ -3800 & -3500 & -2500 \end{bmatrix},$$

$$\boldsymbol{Aeq}=\begin{bmatrix} 1 & 1 & 1 \end{bmatrix},\quad \boldsymbol{beq}=1\quad \boldsymbol{b}=\begin{bmatrix} 70 \\ -2900 \end{bmatrix},\quad \boldsymbol{lb}=\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix},\quad \boldsymbol{ub}=\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

**【程序清单】**

例程 6-3 是求解的 MATLAB 代码。

例程 6-3

```
f=[8400,7500,6700]';
A=[60,65,80; -3800,-3500,-2500];
b=[70,-2900]';
Aeq=[1,1,1];
beq=1;
lb=[0,0,0]';
ub=[1,1,1]';

x0=[0.5,0.5,0.5]';%初始点

[x, fval] = linprog(f,A,b,Aeq,beq,lb,ub)
```

【结果输出】

```
x =
        0.0000
        0.6667
        0.3333
fval =
        6.2333e+003
```

也就是在最优配棉方案中，没有国棉 131，国棉 229 配比为 0.667，国棉 327 配比为 0.333，此时混棉的单价为 7233 元/吨。

# 6.4 优化工具箱在投资领域中的应用

资金最优使用和资金投资优化组合是金融领域中常常遇见的问题，我们可以使用运筹学的优化理论，使得它们产生最大的效益。优化工具箱在这方面也大有用武之地。

## 6.4.1 资金最优使用方案

设有 400 万元资金，要求 4 年内使用完，若在一年内使用资金 $x$ 万元，则可获得效益 $\sqrt{x}$ 万元（效益不能再使用），当年不用的资金可存入银行，年利率为 10%，试制定出这笔资金的实用方案，以使 4 年的经济效益总和为最大。

1. 问题的提出

很明显，针对现有资金 400 万元，对于不同的使用方案，4 年内所获得的效益总和是不相同的。比如第一年就把 400 万元全部用完，则获得效益总和为 20.0 万元；若前三年均不用这笔资金，而是把它存入银行，则第四年时本息和为：$400*(1.1)^3=532.4$（万元），再把它全部用完，则效益总和为 23.07 万元，比第一种方案效益多 3 万多元。所以用最优化的方法可以制定出一种最优的使用方案，使得 4 年的经济效益总和为最大。



2. 问题分析

本问题考虑的是使 4 年的经济效益总和最大，而不是 4 年后的资金总和最大；另一方面，资金如果没有被使用，则不会产生效益。

3. 模型符号约定

$x1$：表示第一年所使用的资金数；
$x2$：表示第二年所使用的资金数；
$x3$：表示第三年所使用的资金数；
$x4$：表示第四年所使用的资金数；
$TOTLE$：表示 4 年的效益总和。

4. 模型的建立

目标函数为：

$$\max TOTLE = \sqrt{x1} + \sqrt{x2} + \sqrt{x3} + \sqrt{x4}$$

满足约束：每一年所使用的资金数既不能为负值，也不能超过当年所拥有的资金数。
第一年：$0 \leqslant x1 \leqslant 400$；
第二年：$0 \leqslant x2 \leqslant (400 - x1) * 1.1$（第一年未使用资金存入银行一年后的本利之和）；
第三年：$0 \leqslant x3 \leqslant [(400 - x1) * 1.1 - x2] * 1.1$
第四年：$0 \leqslant x4 \leqslant \{[(400 - x1) * 1.1 - x2] * 1.1 - x3\} * 1.1$
所以最终资金使用问题的数学模型为：

$$\max TOTLE = \sqrt{x1} + \sqrt{x2} + \sqrt{x3} + \sqrt{x4}$$

$$\text{且满足：}\begin{cases} x1 \leqslant 400 \\ 1.1 * x1 + x2 \leqslant 440 \\ 1.21 * x1 + 1.1 * x2 + x3 \leqslant 484 \\ 1.331 * x1 + 1.21 * x2 + 1.1 * x3 + x4 \leqslant 532.4 \\ x1, x2, x3, x4 \geqslant 0 \end{cases}$$

5. 模型的求解

【计算方法】

由于在 MATLAB 优化工具箱中一般都是求最小值，因此计算时上述目标函数转化为求：

$$\min TOTLE = -\sqrt{x1} - \sqrt{x2} - \sqrt{x3} - \sqrt{x4}$$

由于目标函数是一个非线性函数，我们可以用 MATLAB 优化工具箱中的非线性约束优化函数求解，其形式为：

$$\min TOTLE = -\sqrt{x1} - \sqrt{x2} - \sqrt{x3} - \sqrt{x4}$$

且满足：
$$\begin{cases} \boldsymbol{A} * \boldsymbol{x} \leqslant \boldsymbol{b} \\ \boldsymbol{lb} \leqslant \boldsymbol{x} \leqslant \boldsymbol{ub} \end{cases}$$

其中：

$$\boldsymbol{A} = \begin{bmatrix} 1.1 & 1 & 0 & 0 \\ 1.21 & 1.1 & 1 & 0 \\ 1.331 & 1.21 & 1.1 & 1 \end{bmatrix}, \quad \boldsymbol{x} = \begin{bmatrix} x1 \\ x2 \\ x3 \\ x4 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} 440 \\ 484 \\ 532.4 \end{bmatrix}$$

$$\boldsymbol{lb} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \boldsymbol{ub} = \begin{bmatrix} 400 \\ 1000 \\ 1000 \\ 1000 \end{bmatrix}$$

**【程序清单】**

例程 6-4 是求解的 MATLAB 代码。

例程 6-4

```
%编写目标函数
function y=totle(x)
y=-sqrt(x(1))-sqrt(x(2))-sqrt(x(3))-sqrt(x(4));

A=[1.1,1,0,0;1.21,1.1,1,0;1.331,1.21,1.1,1];
b=[440,484,532.4]';
lb=[0,0,0,0]';
ub=[440,1000,1000,1000]';

x0=[100,100,100,100]';
[x,fval]=fmincon('totle',x0,A,b,[],[],lb,ub)
```

**【结果输出】**

```
x =
        84.2435
        106.6351
        128.9034
        148.2397
fval =
        -43.0821
```

也即如表 6-7 所示。

表 6-7 资金最优使用方案

| | 第一年 | 第二年 | 第三年 | 第四年 |
|---|---|---|---|---|
| 现有资金 | 400 | 356.6 | 284.2 | 182.2 |
| 使用金额 | 84.2 | 106.6 | 128.9 | 148.2 |

4 年效益总和最大为：$TOTLE=\sqrt{84.2}+\sqrt{107.6}+\sqrt{128.9}+\sqrt{148.2}=43.1$，这是 20.0 万元的 2 倍多，这反映出进行定量的优化计算的作用。所以，一些业内人士称最优化方法为“不需要增加投入就能增加产出的手段”。

## 6.4.2 资金投资优化组合决策

市场上有几种资产（如股票、债券等）$S_i$（$i$=1,⋯,$n$）供投资者选择，某公司有数额为 $M$ 万元的一笔相当大的资金可用做一个时期的投资。公司财务分析人员对几种资产进行了评估，估算出在这一时期内购买 $S_i$ 的平均收益率为 $r_i$，并预测出购买 $S_i$ 的风险损失率为 $ri$。考虑到投资越分散，总的风险越小，公司决定，当用这笔资金购买若干种资产时，总体风险可用所投资的 $S_i$ 中最大的一个风险来度量。

购买 $S_i$ 要付交易费，费率为 $p_i$，并且当购买不超过给定值 $u_i$ 时，交易费按购买 $u_i$ 计算（不买当然无需交费），另外，假定同期银行存款利率是 $r_0$（$r_0$=5%），且既无交易费又无风险。

已知 $n$=4 时的相关数据如表 6-8 所示。

表 6-8 投资相关数据

| $S_i$ | $r_i$(%) | $q_i$(%) | $p_i$(%) | $u_i$(%) |
|---|---|---|---|---|
| $S_1$ | 28 | 2.5 | 1 | 103 |
| $S_2$ | 21 | 1.5 | 2 | 198 |
| $S_3$ | 23 | 5.5 | 4.5 | 52 |
| $S_4$ | 25 | 2.6 | 6.5 | 40 |

要求为该公司设计一种最佳投资组合方案，即用给定的资金 $M$，有选择地购买若干种资产或银行生息，使净收益尽可能大，而总体风险尽可能小。

1．问题的提出

资产投资的经济目的就是价值增值，这是投资的效益特性；宏观经济环境和微观经济条件的变化，均会造成投资预期收益的不确定性，这是投资的风险特征。投资风险和预期收益就是资产投资的两大制约因素。组合投资可以分散总体投资风险，若既要使投资风险最小，又要预期收益最大的投资组合，无论在理论上还是实际运作中往往是行不通的，这样投资者必须在投资风险和预期收益两者中做出权衡，找到最佳结合点。

2．问题的分析

这是一个资产的投资组合问题，投资的期望收益和风险是投资者需要综合考虑的两个

目标，为此在建模的过程中引入参数$\mu$表示投资者对期望收益的偏好系数，从而（$1-\mu$）表示投资者对投资风险的偏好系数（即$\mu$越大，说明投资者更看重于收益大小，不敢冒太大的投资风险；$\mu$越小，说明投资者敢于冒大的投资风险），从而可以将两个优化目标合并成一个优化目标，建立一个单目标的非线性规划模型（SNP）来求解最佳投资组合决策方案。

3．模型假设和符号约定

基本假设：

- 公司财务分析人员的资产评估数据可靠；
- 总体风险用所投资的资产中最大的一个风险来度量；
- 足够大；
- 银行存款率 $r_0$=5%；

附加假设：

- 平均收益率=期望收益率/总投资；
- 风险定义为实际因素会给投资者带来的最大可能损失额；
- 风险损失率=风险/总投资；
- 假设只考虑一个投资周期；

符号约定：

- $M$：公司用于投资的总资金额；
- $S_i$：第 $i$ 种资产；
- $N$：资产的个数；
- $r_i$：一个时期内购买的平均收益率；
- $p_i$：购买 $S_i$ 的交易费费率；
- $u_i$：投资 $S_i$ 时的给定值；
- $xi$：对 $S_i$ 的投资额占的比例；
- $RI$：总体风险；
- $ri$：总体风险损失率，$ri=RI/M$；
- $V$：总体收益；
- $v$：总体平均收益。

4．模型的分析

（1）交易费用函数为：

$$F(x)=\begin{cases} 0 & M*x_i=0 \\ u_i*p_i & 0<M*x_i\leqslant u_i \\ M*x_i*p_i & M*x_i\geqslant u_i \end{cases}$$

（2）我们也将银行看做是一种资产，其中 $r_0$=5%，$q_0$=0，$p_0$=0，$u_0$=0。

5．模型的建立

（1）基本模型

记一个投资周期内的净收益为 $V$：

则：　$V=\sum_{i=0}^{n}(M*x_i*r_i-F(x_i))$

组合投资总体风险定义为所投资项目中的最大风险：

即：　$RI=\max_i\{M*x_i*q_i\}$

我们的目标就是使净收益尽可能大，总体风险尽可能小，因此可以建立以下的投资决策数学模型：

$$\max V=\max\left\{\sum_{i=0}^{n}(M*x_i*r_i-F(x_i))\right\}$$

$$\min RI=\min\left\{\max_i M*x_i*q_i\right\}$$

且满足：$\begin{cases}\sum_{i=0}^{n}x_i=1\\0\leqslant x_i\leqslant 1\end{cases}$，$i=0,1,\cdots,n$　　（VNP 模型）

这显然是一个多目标非线性规划（VNP）模型，不好直接求解，可以进行模型转换。

（2）单目标非线性规划模型——偏好系数加权法

根据投资理论，要求风险越小，且同时与其投资收益最大的投资组合无论在理论上还是在实际运作中都是不可能达到的。因此，投资者必须在这两者之间做出权衡，选择投资者对这两方面都比较满意的投资组合。

我们将 VNP 模型中的两个目标函数分别设置权值：设$\mu$和（1-$\mu$）分别为净投资收益和总体投资风险的权重值，于是 VNP 模型就可以转化为单目标非线性规划模型（SNP）。

$$\begin{aligned}\min T&=(1-\mu)*RI-\mu*V\\&=(1-\mu)\max_i\{M*x_i*q_i\}-\mu*\sum_{i=0}^{n}(M*x_i*r_i-F(x_i))\end{aligned}$$

且满足：$\begin{cases}\sum_{i=0}^{n}x_i=1\\0\leqslant x_i\leqslant 1\ ,\ i=0,1,\cdots,n\\\mu\in[0,1]\end{cases}$　　（SNP 模型）

权重数$\mu$和（1-$\mu$）分别表示公司决策者对投资净收益和总体投资风险的重视程度，$\mu$值越大，表示公司决策者更注重于投资净收益，敢于冒风险；反之，表示公司决策者比较保守，不敢冒太大的投资风险。

（3）线性规划模型

事实上引入变量 $t$，SNP 模型可进一步转化：

令 $$t = \max_i\{M * x_i * q_i\}$$

则 SNP 模型变为：

$$\min T = (1-\mu) * t - \mu * \sum_{i=0}^{n} (M * x_i * r_i - F(x_i))$$

且满足：
$$\begin{cases} M * x_i * q_i \leqslant t \\ 0 \leqslant x_i \leqslant 1 \\ \sum_{i=0}^{n} x_i = 1 \\ M * x_j * q_j = t, j \in \{1, 2, \cdots, n\} \end{cases}$$ （SNP'模型）

SNP 模型转化为 $n$ 个含有（$n$+1）个变量的非线性规划模型，且当函数 $F(x_i)$可线性化时，每一个非线性模型又转化为线性规划模型。

6．模型的求解

**【计算方法】**

对于 SNP'模型，我们运用惩罚函数法进行求解。

这是 $n$ 个关于变量（$x_1, x_2, \cdots, x_n$）以及 $t$ 的非线性规划模型，定义惩罚函数：

$$P_j(X) = [\min(0, \min\{x_i\})]^2 + (1 - \sum_{i=0}^{n} x_i)^2$$

其中：$\begin{cases} X = (x_1, x_2, \cdots, x_n) \\ j \in \{1, \cdots, n\} \end{cases}$ （NCP 模型）

从而问题集中于求解 $n$ 个无约束非线性规划问题：

$$\min(H, w) = T + w * P_j(X)$$ ，$w$>0为惩罚因子

**【程序清单】**

例程 6-5 是求解的 MATLAB 代码。

例程 6-5

```
%编写惩罚函数 pen.m
function y=pen(x)
s=0;
for i=1:1:length(x)
    s=s+x(i)
end
```

```
y=min(0,min(x)).^2+(1-s).^2;

%编写交易费用函数 h.m
function y=h(x,u,p)
if (x>u)
    y=x*p
else
    if (x>0)
        y=u*p
    else
        y=0;
    end
end

%编写目标函数 f.m
function y=f(x)
global w;
global M;

r=[0.28,0.21,0.23,0.25,0.05];
q=0.01*[2.5,1.5,5.5,2.6,0];
p=0.01*[1,2,4.5,6.5,0];
u=[103,198,52,40,0];

penalty=1e-25;
t=0;
for i=1:1:4
    t=t+h(x(i)*M,u(i),p(i));
end

t=t-M*sum(r.*x);
y=w*t-(1-w)*max(x.*q)*M+penalty*pen(x);

for k=1:1:11
    w=(k-1)*0.1;
    s=5000;
    for j=1:1:50
        c=rand(1,5);
        d=sum(c);
        x0=c/d;
        x=fminsearch('f',x0);
        if(f(x)<s)
            s=f(x);
            xc=x;
        end
    end
    t=0;
```

```
        for i=1:1:4
            t=t+h(xc(i)*M,u(i),p(i));
        end
            FR(k)=f(xc);
        net(k)=t+M*sum(r.*xc);
        risk(k)=M*max(xc.*q);
        xn(k,:)=xc;
    end
    figure(1)
    plot(w,xn(:,1),'*',w,xn(:,2),'c',w,xn(:,3),'s',w,xn(:,4),'v',w,xn(:,5),'0');
```

**【结果输出】**

在计算过程中，我们让 $M$ 的值从小依次增大时发现，随着 $M$ 值增大，投资组合随之发生变化，但当 $M$ 值大于 2 000 万元时，投资组合不再发生变化，分别取 $M$=5 000 万元和 $M$=1 000 万元得到以下结果.

（1）M=5 000 万元

表 6-9 列出了此时的投资组合，并且计算出对应的总体平均收益率 $V$ 和总体平均风险率 $ri$.

（2）M=1000 万元

表 6-10 列出了此时的投资组合，并且计算出对应的总体平均收益率 $V$ 和总体平均风险率 $ri$。

表 6-9　不同$\mu$值对应的最佳投资组合（$M$ 较大时 $M$=5 000 万元）

| $\mu$ | x0 | x1 | x2 | x3 | x4 | ri | v |
|---|---|---|---|---|---|---|---|
| 0.00 | 1.000 | 0.000 | 0.000 | 0.000 | 0.000 | 5% | 0 |
| 0.10 | 0.000 | 0.245 | 0.408 | 0.111 | 0.236 | 20.8% | 0.61% |
| 0.22 | 0.000 | 0.375 | 0.625 | 0.000 | 0.000 | 22% | 0.94% |
| 0.50 | 0.000 | 1.000 | 0.000 | 0.000 | 0.000 | 27% | 2.5% |
| 1.00 | 0.000 | 1.000 | 0.000 | 0.000 | 0.000 | 27% | 2.5% |

表 6-10　不同$\mu$值对应的最佳投资组合（$M$ 较小时 $M$=1 000 万元）

| $\mu$ | x0 | x1 | x2 | x3 | x4 | ri | v |
|---|---|---|---|---|---|---|---|
| 0.00 | 1.000 | 0.000 | 0.000 | 0.000 | 0.000 | 5% | 0 |
| 0.10 | 0.000 | 0.320 | 0.534 | 0.146 | 0.000 | 21.5% | 0.8% |
| 0.20 | 0.000 | 0.375 | 0.625 | 0.000 | 0.000 | 22% | 0.94% |
| 0.24 | 0.000 | 1.000 | 0.000 | 0.000 | 0.000 | 27% | 3.4% |
| 1.00 | 0.000 | 1.000 | 0.000 | 0.000 | 0.000 | 27% | 2.5% |

# 6.5　优化工具箱在优化设计中的应用

所谓"优化设计"就是根据给定的设计要求和现有的工程技术条件，应用专业理论和

优化方法，通过计算机从满足给定设计要求的许多可行方案中，按照规定的性能指标（目标函数）自动选出最优的设计方案。优化设计与传统的工程设计不同：传统工程设计根据经验和判断先设计制定设计方案，然后对给定方案进行分析和校核，即“系统分析”；而优化设计综合各方面的因素（包括设计要求、性能指标、约束条件等），应用优化理论及方法，产生一个最优设计方案，即“系统综合”。

优化设计的过程从输入一个初始方案（一组设计变量的初值）开始，经过数学模型的分析和用优化方法进行自动优选，最后得到一个最优的设计方案，其流程图如图 6-1 所示。在挑选设计变量时，应尽量减少设计变量的个数，以降低优化设计的维数，并使目标函数和约束条件尽可能简单，一般对变量的处理方法包括：对变量进行适当的归并；对变量进行形式上的变换等。不同类型的最优化问题可以选择不同类型的目标函数，但至少应选择足以表征系统优劣的某个或某些特征指标。为了寻优方便，目标函数最好尽可能表达成线性函数或二次型函数；对于非线性函数或非二次型函数的目标函数，也可通过变量变换、函数变换或泰勒技术展开等方法表达成线性函数或二次型函数。

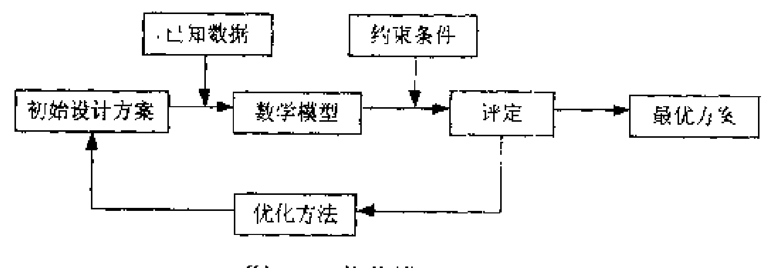



图 6-1　优化设计流程图

下面介绍一个直流电动机跟踪系统补偿校正的优化设计例子。

由直流电动机组成的输入跟踪系统示意图如图 6-2 所示。图中 $G(s)$为直流电动机的传递函数，$G_c(s)$为超前校正的补偿校正器的传递函数，它们分别为：

$$G(s)=\frac{25}{s(s+20)(s+1)}$$

$$G_c(s)=\frac{A(s+z)}{1+s/p}$$

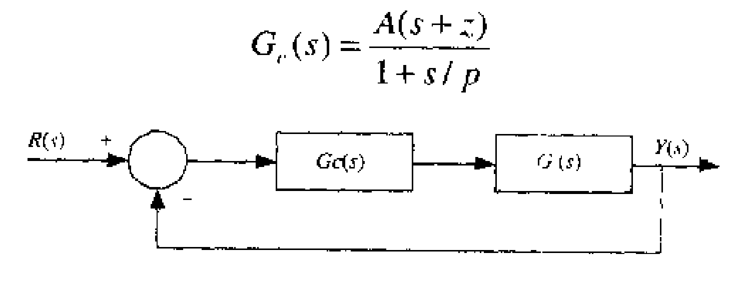



图 6-2　直流电动机输入跟踪系统示意图

为了实现输出对输入的跟踪，系统中采用了单位反馈组成闭环控制系统，现要求设计超前步长校正器，使得输出能跟踪输入。

**1、问题的分析**

由题意，$s=-z$ 和 $s=-p$ 分别为步长校正器的零点和极点，由经典控制理论可知，为了

实现超前补偿，必须满足 $z<<p$ 且极点$-p$ 必须配置在被控对象极点的左侧且远离对象的极点，故值 $p$ 很大，此时超前补偿校正器的传递函数可近似表达为：

$$G_c(s) \approx A(s+z)$$

此时系统的前向通道的传递函数为：

$$G(s)G_c(s) = \frac{25A(s+z)}{s(s+20)(s+1)}$$

规定一组状态变量，令 x1=y。系统的模拟结构图如图 6-3 所示，系统状态方程为：

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = 25A(r - x_1 + x_3) - 20x_2 \\ \dot{x}_3 = (z-1)(r - x_1) - x_3 \end{cases}$$

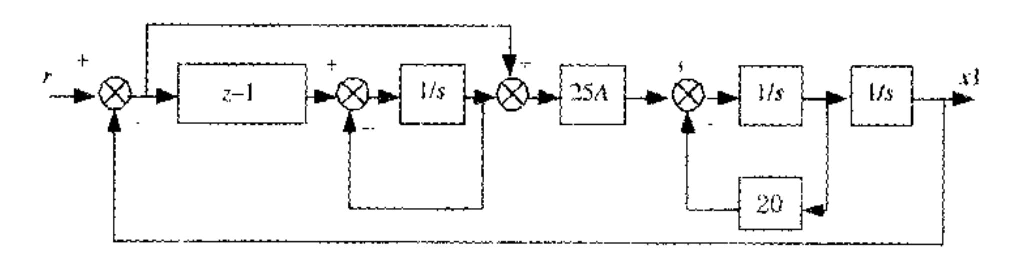



图 6-3　系统的模拟结构图

为了使跟踪系统具有良好的动态性能指标，则要求误差信号 $e=r-x_1$ 尽可能小，故可取平方误差积分准则作为目标函数，即：

$$J = \int_{t_0}^{t_f} e^2 \mathrm{d}t = \int_{t_0}^{t_f} (r - x_1)^2 \mathrm{d}t$$

显然本跟踪系统的参数最优化问题可归纳为选取参数 $A$ 和 $z$，使跟踪误差 $e$ 平方的积分为最小。

2、模型的建立

为了进行优化设计，系统的输入信号 $r(t)$可以是给定的。根据经典控制理论可知，对于线性系统，如果系统对单位阶跃响应是满意的，则一般对任何输入的响应都会是较满意的。所以，系统的输入信号 $r(t)$可选取单位阶跃的输入函数，即：

$$r(t) = u(t) = \begin{cases} 1 & t \ge 0 \\ 0 & t < 0 \end{cases}$$

为了保证系统趋于稳态，目标函数中的初始时间可取为 $t_0=0$，而终端时间 $t_f$ 必须选得

足够大。另外，根据系统要求，规定终端状态 $x_1(t_f)=1$，$x_1(t_f)=x_2(t_f)=0$。因此，应用惩罚函数法得到目标函数为：

$$\tilde{J}=\int_0^{t_f}(1-x_1)^2\mathrm{d}t+s_1[1-x_1(t_f)]^2+s_2[x_2(t_f)]^2$$

式中，$s_1$ 和 $s_2$ 为权系数。

哈密尔顿函数为：

$$H=(1-x_1)^2+\lambda_1x_2+\lambda_2[25A(1-x_1+x_3)-20x_2]+\lambda_3[(z-1)(1-x_1)-x_3]$$

状态方程为：

$$\begin{cases}\dot{\lambda}_1=2(1-x_1)+25A\lambda_2+(z-1)\lambda_3\\ \dot{\lambda}_2=-\lambda_1+20\lambda_3\\ \dot{\lambda}_3=-25A\lambda_2+\lambda_3\end{cases}$$

由目标函数 $\tilde{J}$ 的终端项可得：

$$\begin{cases}\lambda_1(t_f)=2s_1[x_1(t_f)-1]\\ \lambda_2(t_f)=2s_2x_2(t_f)\\ \lambda_3(t_f)=0\end{cases}$$

为了求解参数最优化问题，取控制 $u=[A, z]^T$。根据参数最优化的梯度定义，则有：

$$g[u]=\nabla\tilde{J}(u)=\int_{t_0}^{t_f}(\frac{\partial H}{\partial u})^T\mathrm{d}t=\int_{t_0}^{t_f}[\frac{\partial H}{\partial A}\quad\frac{\partial H}{\partial z}]^T\mathrm{d}t$$

即：

$$g\begin{bmatrix}A\\ z\end{bmatrix}=\begin{bmatrix}\int_0^{t_f}25[x_3(t)-x_1(t)+1]\lambda_2(t)\mathrm{d}t\\ \int_0^{t_f}[1-x_1(t)]\lambda_3(t)\mathrm{d}t\end{bmatrix}$$

因此，本问题的数学模型为求解 $A$ 和 $u$，使得

$$\min\tilde{J}=\min(\int_0^{t_f}(1-x_1)^2\mathrm{d}t+s_1[1-x_1(t_f)]^2+s_2[x_2(t_f)]^2)$$

显然这是一个无约束优化问题。

3．模型的求解

**【计算方法】**

（1）惩罚因子 $s_1$，$s_2$ 的选取

利用惩罚函数法将有终端约束的最优化问题转化为无约束的最优化问题，从后者得到

的解是前者解的近似解。显然，目标函数中的惩罚因子 $s_1,s_2$ 取值越大，则所得到的最优解对于终端约束越精确，但却偏离终端约束条件下原目标函数的最优解。这里选取 $s_1$=1 000，$s_2$=1 000。

（2）初始值的选取

根据经典控制理论的设计方法，被控对象在 $s$=-1 处的主导极点应与超前补偿校正器的零点 $s$=-$z$ 相抵消，即 $z$=1。此时，系数的闭环传递函数可写成：

$$G_R(s)=\frac{Y(s)}{R(s)}=\frac{25A}{s(s+20)+25A}=\frac{25A}{s^2+20s+25A}$$

与二阶系统的标准闭环传递函数表达式相比较，则有：

$$\omega_n^{\ 2}=25A,\qquad 2\zeta\omega_n=20$$

如果选取阻尼系数ζ=0.7，则无阻尼自然振荡频率 $\omega_n$=14.29，$A$=8.163。因此取参数初始值为：

$$\begin{bmatrix}A^{(0)}\\ z^{(0)}\end{bmatrix}=\begin{bmatrix}8.163\\ 1\end{bmatrix}$$

（3）终端时间 $t_f$ 的选择

终端时间 $t_f$ 必须选得足够大，以保证系统趋于稳态。一般，可取终端时间 $t_f$ 等于单位阶跃响应的调整时间 $t_s$。由经典控制理论可知，调整时间 $t_s$ 近似等于系统时间常数 $T$ 的 4 倍，即：

$t_s$=4$T$=0.4s，　这里取 $t_f$=0.5s

**【程序清单】**

例程 6-6 是求解的 MATLAB 代码。

例程 6-6

```
%S-函数 trackoptim.mdl 如图 6-4 所示。
```

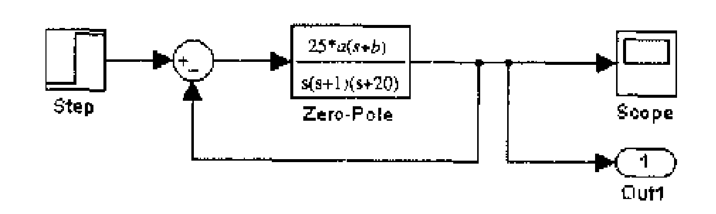



图 6-4　S-函数 trackoptim.mdl

```
%目标函数
function F = tracklsq(input)
a=input(1);
b=input(2);
opt = simset('solver','ode5','SrcWorkspace','Current');
```

```
[tout,xout,yout] = sim('trackoptim',[0 5],opt);
%计算误差信号
F = yout-1;
%调用优化函数
&调入仿真模型
trackoptim
%初始化变量
input=[8.163 1];
%设置优化参数
options = optimset('LargeScale','off','Display','iter', 'TolX',0.001,'TolFun',0.001);
%调用优化函数
[output,fval]=lsqnonlin('tracklsq',input,[],[],options)
% 输出参数
a=output(1)
b=output(2)
```

**【结果输出】**

迭代过程如表 6-11 所示。

表 6-11　迭代过程

| Iteration | Func-count | Residual | Step-size | Directional derivative | Lambda |
|---|---|---|---|---|---|
| 1 | 2 | 11.1816 | 1 | -0.2 | |
| 2 | 8 | 11.0628 | 1 | -5.26e-005 | 5.54035 |
| 3 | 21 | 11.0552 | 227 | -1.38e-006 | 0.0225184 |

```
output =
        12.7085     1.1298
fval =
        11.0496
a =     12.7085
b =     1.1298
```

系统预定输入阶跃信号如图 6-5 所示。

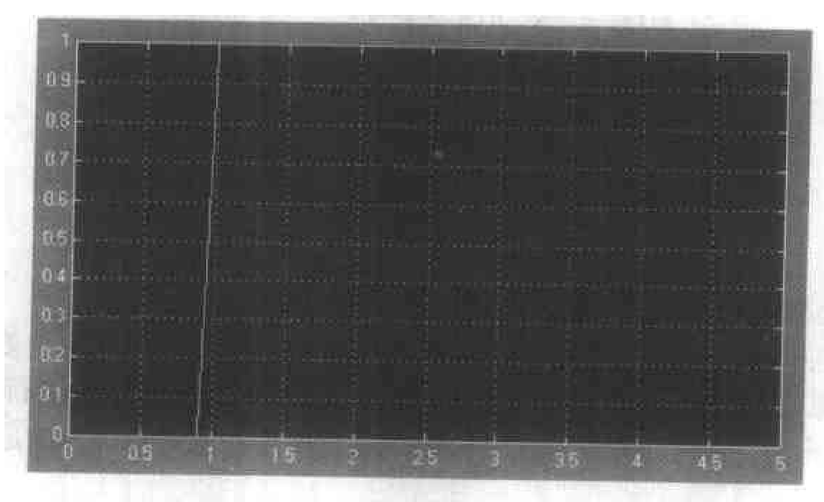


图 6-5　输入的阶跃信号

优化后系统的阶跃响应如图 6-6 所示。

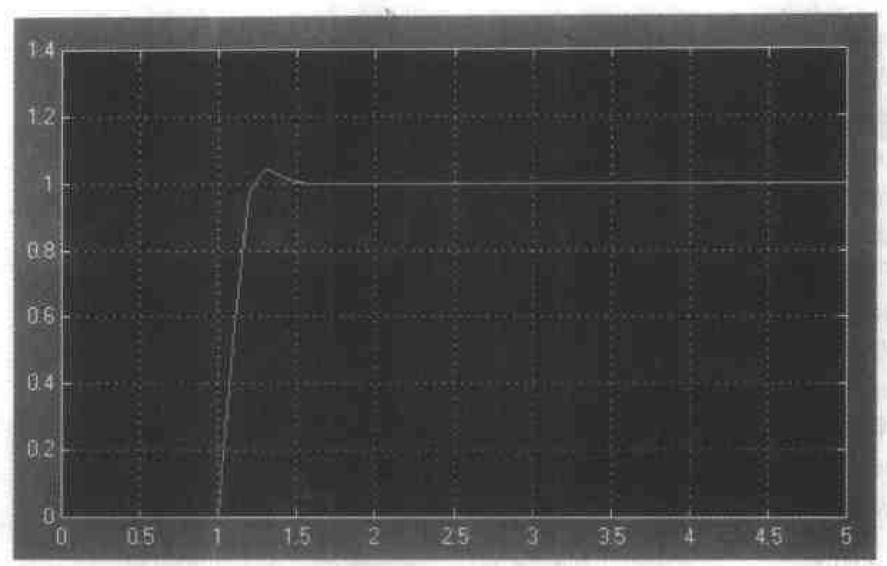



图 6-6　参数优化设计后的阶跃响应

比较两图可以看出对参数 $a$ 和 $b$ 进行优化设计，可以满足系统要求。

# 6.6　优化工具箱在信号处理中的应用

本节将介绍一个 FIR 滤波器的优化设计例子。

1．问题的提出

考虑设计一个线性相位的有限冲击响应滤波器，现在要求设计一个低通滤波器：当 $0 \leqslant \omega \leqslant 0.1\text{Hz}$ 时，滤波器幅值为 1；当 $0.15\text{Hz} \leqslant \omega \leqslant 0.5\text{Hz}$ 时，滤波器幅值为 0。

2．问题的分析

有限冲击响应滤波器的频率响应 $H(f)$ 由下式得到：

$$H(f) = \sum_{n=0}^{2M} h(n) e^{-j2\pi fn}$$

$$= A(f) e^{-j2\pi fM}$$

$$A(f) = \sum_{n=0}^{M-1} a(n)\cos(2\pi fn)$$

其中 $A(f)$是频率响应的幅值函数。

解决此问题的一种方法是应用多目标逼近方法来满足频率响应的幅值要求，假设给定计算频率响应幅值的函数，则可以应用 MATLAB 优化工具箱的 fgoalattain 函数来改变 FIR 滤波器的系数 $a(n)$，使得设计出来的滤波器的幅值响应在允许范围内满足预定的设计要求。

### 3．问题的求解

**【计算方法】**

为了能够构造出一个多目标逼近问题，我们首先必须确定目标值 goal 和权重 weight。本问题中，当频率位于 0.15Hz～0.5Hz 时，目标值为 1；当频率位于 0.1Hz～0.15Hz 时，目标值为 0；而当频率位于 0Hz～0.1Hz 时，没有给出目标值，因而在计算时，这一区间没有目标值和权重值。

**【程序清单】**

例程 6-7 是求解的 MATLAB 代码。

例程 6-7

```
%Step 1：计算幅值
function y = filtmin(a,w)
n = length(a);
y = cos(w'*(0:n-1)*2*pi)*a ;

%Step 2：调用优化函数
% 绘制初始系数的滤波器频响图
a0 = ones(15,1);
incr = 50;
w = linspace(0,0.5,incr);

y0 = filtmin(a0,w);
clf, plot(w,y0,'-:');
drawnow;

% 构造多目标逼近问题
w1 = linspace(0,0.1,incr) ;
w2 = linspace(0.15,0.5,incr);
w0 = [w1 w2];
goal = [1.0*ones(1,length(w1)) zeros(1,length(w2))];
weight = ones(size(goal));

% 调用 fgoalattain 函数
options = optimset('GoalsExactAchieve',length(goal));
[a,fval,attainfactor,exitflag] = fgoalattain(@filtmin,...
   a0,goal,weight,[],[],[],[],[],[],[],options,w0);
% 绘制优化系数的滤波器频响图
y = filtmin(a,w);
hold on, plot(w,y,'r')
axis([0 0.5 -3 3])
xlabel('Frequency (Hz)')
ylabel('Magnitude Response (dB)')
legend('initial', 'final')
```

```
grid on
```



程序中目标值通过输入参数 goal 传递给 fgoalattain 函数，goal 的长度与函数 filtmin 返回值的长度相同，因而目标值能够同时得到满足；另外，权重 weight 取 goal 的绝对值。

【结果输出】

FIR 滤波器的系数为：

```
a =
    0.2512
    0.4509
    0.3158
    0.1471
    0.0019
   -0.0801
   -0.0908
   -0.0515
    0.0026
    0.0400
    0.0466
    0.0284
    0.0031
   -0.0149
   -0.0364
```

采样点的幅值如表 6-12 所示。

表 6-12　采样点的幅值

| 1.0141 | 1.0147 | 1.0156 | 1.0169 | 1.0183 | 1.0199 | 1.0215 | 1.0230 |
|---|---|---|---|---|---|---|---|
| 1.0243 | 1.0253 | 1.0259 | 1.0260 | 1.0255 | 1.0244 | 1.0226 | 1.0202 |
| 1.0172 | 1.0136 | 1.0095 | 1.0051 | 1.0004 | 0.9957 | 0.9910 | 0.9866 |
| 0.9826 | 0.9792 | 0.9766 | 0.9748 | 0.9740 | 0.9742 | 0.9756 | 0.9779 |
| 0.9813 | 0.9907 | 0.9963 | 0.9856 | 1.0022 | 1.0081 | 1.0137 | 1.0187 |
| 1.0227 | 1.0252 | 1.0260 | 1.0246 | 1.0207 | 1.0140 | 1.0041 | 0.9908 |
| 0.9740 | 0.0260 | -0.0187 | -0.0260 | 0.0106 | 0.0108 | 0.0252 | 0.0260 |
| 0.0140 | -0.0041 | -0.0197 | -0.0260 | -0.0205 | -0.0059 | 0.0112 | 0.0236 |
| 0.0260 | 0.0175 | 0.0017 | -0.0149 | -0.0256 | -0.0260 | -0.0160 | 0.0003 |
| -0.0242 | 0.0164 | 0.0260 | 0.0252 | 0.0145 | -0.0019 | -0.0175 | -0.0260 |
| -0.0193 | -0.0129 | 0.0035 | 0.0184 | 0.0260 | 0.0233 | 0.0114 | -0.0049 |
| -0.0260 | -0.0225 | -0.0101 | 0.0063 | 0.0201 | 0.0260 | 0.0217 | 0.0088 |
| -0.0075 | -0.0209 | -0.0260 | | | | | |

初始滤波器和优化后的滤波器的幅值响应如图 6-7 所示。

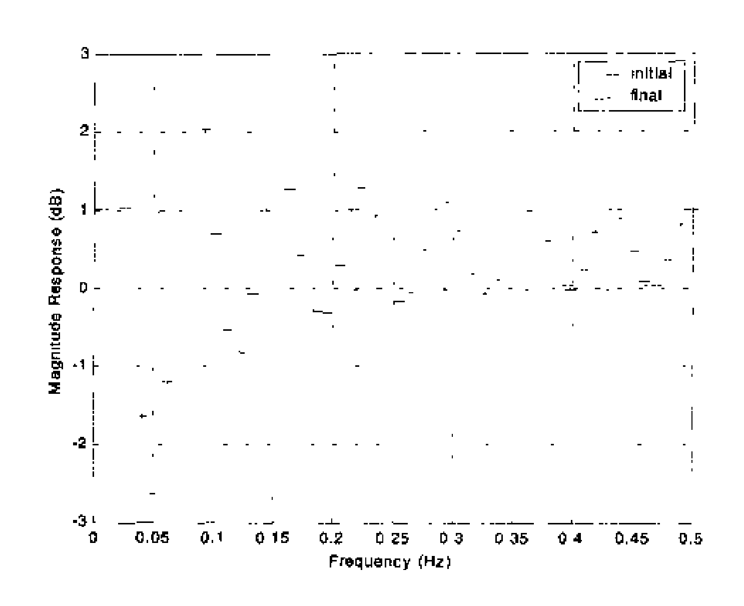



图 6-7　优化后的滤波器幅值响应

可见在一定的误差范围内，设计出来的滤波器满足设计要求。

# 6.7　优化工具箱在生物代谢分析中的应用

生物化学系统是一个非常复杂的网络系统，理解完整的生化系统功能的中心问题是分子间相互作用的大量非线性特性。传统的直觉和试差法难以解决它，必须要用系统的数学模型和分析方法。

## 6.7.1　生物代谢网络优化

现在常用的网络分析方法有代谢控制分析（MCA）和生化系统理论（BST）。在 MCA 中定义了弹性系数、控制系数等重要的表征系数。由控制系数的大小比较即可确定代谢网络中的限制性步骤和关键酶，从而为代谢工程方法提供了目的位点。生化系统理论则是基于模型的方法，将代谢物浓度的变化用生成速率和消耗速率两部分表示，并以幂律函数的形式对各速率与其影响因素（包括中间代谢物和催化各步反应的酶）进行线性化处理，通过在一定约束空间范围内搜索模型的最优解，可以找出系统的最佳影响因素浓度分布。以上两点都存在明显的缺点。在 MCA 中，控制系数式系统在稳态附近的局部特征。在 BST 中，幂律函数的动力学级次数是定义在原始稳态基础之上的，当系统偏离原始稳态点时，两者的准确性都会显著降低。完整生化系统的各种调控机理只有用非线性模型本身才能表达出来，因此需要用非线性模型方法对代谢网络进行优化。

**1．问题的提出**

在代谢网络中，分支路径是最基本的非线性结构，同时也是构成整个复杂代谢网络的基础，考虑如图 6-8 所示的分支代谢路径。

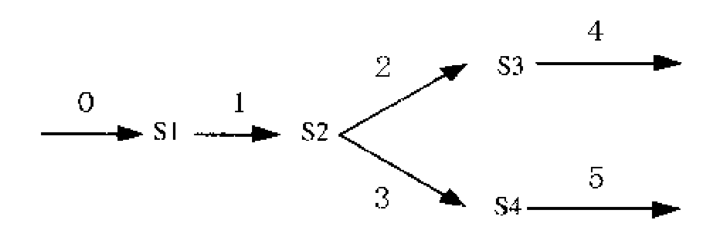



图 6-8　分支代谢路径

有：

$$r_0 = 3$$

$$r_1 = 10 * \frac{s_1}{0.3333 + s_1}$$

$$r_2 = 4.1667 * \frac{s_2}{0.6667 * (1 + \frac{s_3}{1 + 0.5 * s_1}) + s_2}$$

$$r_3 = 7.5 * \frac{s_2}{0.6429 * (1 + \frac{s_4}{7 * (1 + 0.6667 * s_3)}) + s_2 * (1 + \frac{s_4}{3.5 + 2.3333 * s_3})}$$

$$r_4 = 6.25 * \frac{s_3}{0.1875 + s_3}$$

$$r_5 = 3.516 * \frac{s_4}{0.3333 + s_4}$$

各代谢物浓度变化速率方程可写为：

$$\frac{ds_1}{dt} = r_0 - r_1$$

$$\frac{ds_2}{dt} = r_1 - r_2 - r_3$$

$$\frac{ds_3}{dt} = r_2 - r_4$$

$$\frac{ds_4}{dt} = r_3 - r_5$$

通过解非线性方程组 $\frac{ds}{dt} = 0$ 可以得到系统的定解：

0.1428　0.1781　0.2999　0.5187

并假设酶浓度与速率方程成简单的正比例关系：

酶浓度的变化范围是（0.2~5.0）；

中间物浓度变化范围是（-10% ~ +10%）。

现在要求对该分支路径进行优化，目的是使代谢中间物 $S_3$ 得到积累。

**2．模型的分析与建立**

要使代谢中间物 $S_3$ 得到积累，也即使得 $\frac{s_3}{s_3+s_4}$ 最大，从而目标函数为：

$$\max \frac{s_3}{s_3+s_4}$$

引入催化各步反应的酶的浓度为 $x(5),x(6),x(7),x(8),x(9)$。则问题的优化模型为：

$$\max \frac{s_3}{s_3+s_4}$$

且满足：

$$\begin{aligned}
&r_0 - r_1 * x(5) = 0 \\
&r_1 * x(5) - r_2 * x(6) - r_3 * x(7) = 0 \\
&r_2 * x(6) - r_4 * x(9) = 0 \\
&r_3 * x(7) - x_5 * x(9) = 0 \\
&0.1428 * 0.9 \leqslant s_1 \leqslant 0.1428 * 1.1 \\
&0.1781 * 0.9 \leqslant s_2 \leqslant 0.1781 * 1.1 \\
&0.2999 * 0.9 \leqslant s_3 \leqslant 0.2999 * 1.1 \\
&0.5187 * 0.9 \leqslant s_4 \leqslant 0.5187 * 1.1 \\
&0.2 \leqslant x(5), x(6), x(7), x(8), x(9) \leqslant 5
\end{aligned}$$

显然这是一个非线性约束优化问题，可以用 MATLAB 优化工具箱中的 fmincon 函数来求解。

**3．模型的求解**

**【计算方法】**

此模型的优化变量为：

$$\boldsymbol{x} = [s_1, s_2, s_3, s_4, x(5), x(6), x(7), x(8), x(9)]$$

按照函数 fmincon 的用法形式，目标函数为：

$$\min_x f(\boldsymbol{x}) = \min_x \left(\frac{s_3}{s_3+s_4}\right)$$

优化变量满足以下约束：

$$\begin{cases} \boldsymbol{Ceq} * \boldsymbol{x} = 0 \\ \boldsymbol{lb} \leqslant \boldsymbol{x} \leqslant \boldsymbol{ub} \end{cases}$$

其中：$\boldsymbol{lb} = [0.1285, 0.1603, 0.2699, 0.4668, 0.2, 0.2, 0.2, 0.2, 0.2]'$

$\boldsymbol{ub} = [0.1571, 0.1959, 0.3299, 0.5706, 5, 5, 5, 5, 5]$

目标函数 $f(\boldsymbol{x})$和 $\boldsymbol{Ceq}$ 均是非线性方程。

**【程序清单】**

例程 6-8 是求解的 MATLAB 代码。

例程 6-8

```
%编写目标函数
function y=target67(x)
y=-x(3)/(x(3)+x(4));

%编写非线性约束函数
function [c,ceq]=fun67(x)
%无非线性不等式约束
c=[];

r0=3;
r1=10*x(1)/(0.3333+x(1));
r2=4.1667*x(2)/(0.6667*(1+x(3)/(1+0.5*x(1)))+x(2));
r3=6.5*x(2)/(0.6429*(1+x(4)/(7*(1+0.6667*x(3))))+...
     x(2)*(1+x(4)/(3.5+2.3333*x(3)));
r4=6.25*x(3)/(0.1875+x(3));
r5=3.516*x(4)/(0.3333+x(4));
%非线性等式约束
ceq(1)=r0-r1*x(5);
ceq(2)=r1*x(5)-r2*x(6)-r3*x(7);
ceq(3)=r2*x(6)-r4*x(9);
ceq(4)=r3*x(7)-r5*x(9);

%优化变量边界约束
lb=[0.1285 0.1603 0.2699 0.4668 0.2 0.2 0.2 0.2 0.2]';
ub=[0.1571 0.1959 0.3199 0.5706 5.0 5.0 5.0 5.0 5.0]';
%无限型约束
A=[];
b=[];
Aeq=[];
beq=[];
%设定初始值
x0=[0.1428 0.1781 0.2999 0.5187 0.5 0.5 0.5 0.5 0.5]';
%调用优化函数
```



```
[x,fval]=fmincon('target67',x0,A,b,Aeq,beq,lb,ub,'fun67')
```

【结果输出】

```
x=
   0.1571   0.1959   4.1267   0.4668   0.9366   5.0000   0.7369   0.2000   0.8796

fval=
   -0.8984
```

也即：

中间物浓度分别为：$s_1$=0.1571　$s_2$=0.1959　$s_3$=4.1267　$s_4$=0.4668

酶浓度分别为：0.9366　5.0000　0.7369　0.2000　0.8796

由此可见，模型的目标产物 $s_3$ 浓度达到 4.1267，是初始态的 14 倍。

### 6.7.2　确定微生物反应代谢途径

在生化反应中，微生物一般需经多种中间体向代谢产物转化。但在转化过程中，代谢途径却是错综复杂、多种途径并存的。最终除生成目的产物外还会生成多种副产物。因此，若能获知主要的代谢途径，对了解支配产物合成的控制机构、培育优秀的变异株及建立微生物反应速率的数学模型具有重要意义。为了判断主要的代谢途径，传统的做法是采用体外测定有关酶（群）活性的生物化学方法。然而，从另一方面考虑，根据代谢物、产物及副产物的物料平衡也能做出概括性的判断。用物料平衡进行解析，可以在多种可能的代谢途径中获得最可能发生的途径。因为这种推断不能与生物化学（酶学）的知识产生矛盾，所以生物化学法与物料平衡法二者是互补的。

在生物体内进行的生物化学过程都存在着一定的约束条件，因此生物体内的代谢途径的确定就可能归结为有约束条件下的最优化问题。由数学上的最优化计算方法即可得到微生物代谢的最优途径。

**1．问题的提出**

以黑曲菌酵解葡萄糖生产柠檬酸反应为例。分批操作时，在反应后期几乎没有菌体生成，被消耗的葡萄糖几乎全部变为柠檬酸与副产物（草酸与 $CO_2$）。柠檬酸生成的代谢途径如图 6-9 所示。图中标记着各个代谢中间体的摩尔反应速率 $x_1$～$x_{11}$。假定从葡萄糖经糖酵解系统进入代谢途径的丙酮酸（摩尔）消耗速率（是葡萄糖消耗速率的 2 倍）为 $a$，柠檬酸、草酸、$CO_2$ 的摩尔生成速率为 $x_{12}$～$x_{14}$，则下列碳的物料平衡式是成立的：

$$6x_{12}+2x_{13}+x_{14}=3a$$

全部中间体的物料平衡式为：

丙酮酸：$x_1+x_2=a$

草酰乙酸：$x_1-x_3-x_4-x_5=0$

醋酸：$x_2+x_3-x_4-x_7=0$

苹果酸：$x_5-x_6+x_7=0$

琥珀酸：$x_6+x_9+x_{10}=0$

草酸：$x_3 - x_8 - x_{13} = 0$

乙醛酸：$x_8 + x_9 - x_7 = 0$

异柠檬酸：$x_{11} - x_9 - x_{10} = 0$

柠檬酸：$x_4 - x_{11} - x_{12} = 0$

$CO_2$：$2x_{10} - x_1 + x_2 - x_{14} = 0$

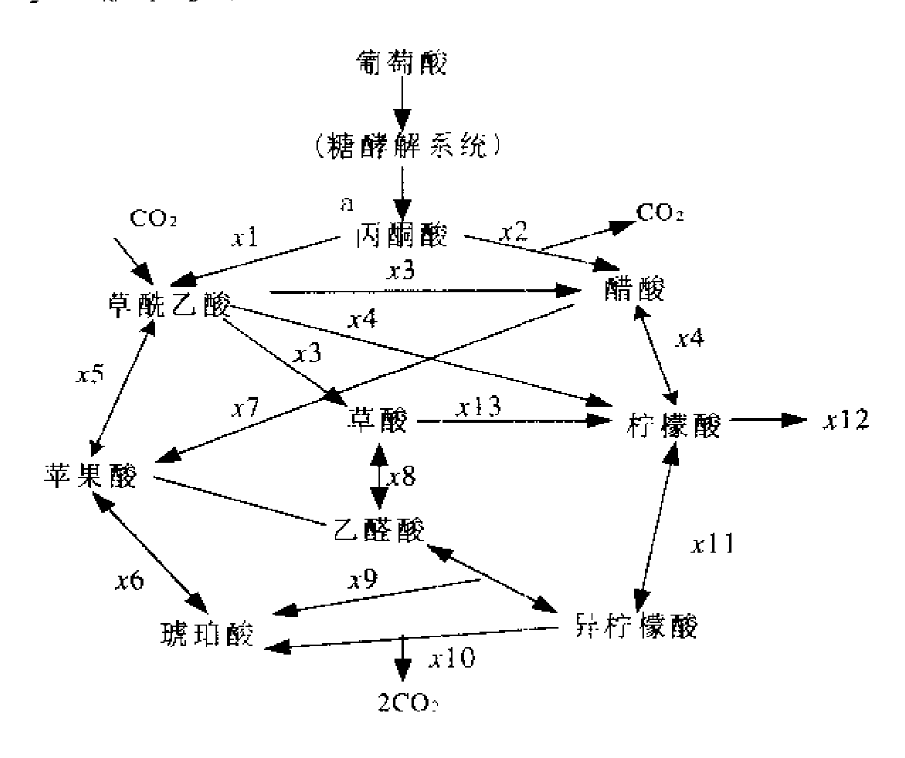



图 6-9　代谢中间体的摩尔反应速率

同时，下列不等式必须成立：

$$x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, x_7 \geq 0, x_{10} \geq 0$$

在利用黑曲菌生成柠檬酸的反应过程中，最终目的就是确定最优代谢途径，使得柠檬酸的产量最大。

**2．模型的建立**

问题的最终目的就是选择一条最优代谢途径，使得柠檬酸的产量最大。柠檬酸的产量表示为：

$$f(\boldsymbol{x}) = 6 * x_{12}$$

根据问题的分析，可以建立如下模型：

$$\max_{x} f(\boldsymbol{x}) = \max(6 * x_{12})$$

满足：

$$
\begin{aligned}
&6*x_{12}+2*x_{13}+x_{14}=3a\\
&x_1+x_2=a\\
&x_1-x_3-x_4-x_5=0\\
&x_2+x_3-x_4-x_7=0\\
&x_5-x_6+x_7=0\\
&x_6+x_9+x_{10}=0\\
&x_3-x_8-x_{13}=0\\
&x_8+x_9-x_7=0\\
&x_{11}-x_9-x_{10}=0\\
&x_4-x_{11}-x_{12}=0\\
&2x_{10}-x_1+x_2-x_{14}=0\\
&x_1,x_2,x_3,x_7,x_{10}\geqslant 0
\end{aligned}
$$

显然，这构成了一个约束线性规划问题。

3．**模型的求解**

**【计算方法】**

将目标函数转化为：

$$\min_{x}-f(\boldsymbol{x})=\min(-6*x_{12})=f^T\boldsymbol{x}$$

约束为：

$$\begin{cases}\boldsymbol{Aeq}*\boldsymbol{x}=\boldsymbol{beq}\\ \boldsymbol{lb}\leqslant\boldsymbol{x}\leqslant\boldsymbol{ub}\end{cases}$$

其中：$\boldsymbol{x}=[x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9,x_{10},x_{11},x_{12},x_{13},x_{14}]^T$，从而可以用 MATLAB 优化工具箱函数 linprog 来解此线性规划问题。

**【程序清单】**

例程 6-9 是求解的 MATLAB 代码。

例程 6-9

```
%参数 a
a=1;
%目标函数系数
f=[0 0 0 0 0 0 0 0 0 0 0 -6 0 0]';
%等式约束矩阵系数
Aeq=[0 0 0 0 0 0 0 0 0 0 0 6 2 1;
```

```
        1 1 0 0 0 0 0 0 0 0 0 0 0 0;
        1 0 -1 -1 -1 0 0 0 0 0 0 0 0 0;
        0 1 1 -1 0 0 -1 0 0 0 0 0 0 0;
        0 0 0 0 1 -1 1 0 0 0 0 0 0 0;
        0 0 0 0 0 1 0 0 1 1 0 0 0 0;
        0 0 1 0 0 0 0 -1 0 0 0 0 -1 0;
        0 0 0 0 0 0 -1 1 1 0 0 0 0 0;
        0 0 0 0 0 0 0 0 -1 -1 -1 0 0 0;
        0 0 0 1 0 0 0 0 0 0 -1 -1 0 0;
        -1 1 0 0 0 0 0 0 0 2 0 0 0 0];
beq=[3*a a 0 0 0 0 0 0 0 0 0]';
%无不等式约束
A=[];
b=[];
%边界约束
lb=[0 0 0 0 0 0 0 0 0 0 0 0 0 0]';
ub=[];
%初始点
x0=[0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 ]';
%调用函数 linprog
[x,fval]=linprog(f,A,b,Aeq,beq,lb,ub,x0)
```

**【结果输出】**

```
x =
    0.5000
    0.5000
    0.0000
    0.5000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.5000
    0.0000
    0.0000
fval =
    -3.0000
```

即 $x_1=0.5$，$x_2=0.5$，$x_4=0.5$，$x_{12}=0.5$

在此条件下柠檬酸生成的最理想途径如图 6-10 所示。

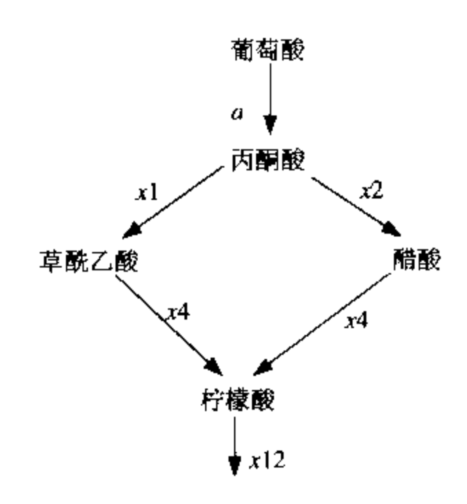



图 6-10 柠檬酸生成的最理想途径

# 6.8 优化工具箱在大规模规划中的应用

第 4 章从理论层面上讲述了大规模规划中的优化算法和 MATLAB 实现，本节将继续应用优化工具箱举出一些综合性的工程应用实例。

## 6.8.1 分子构造问题

### 1．问题的提出

此问题采用大规模算法解决二维分子构造问题，也用于人造卫星的范围与距离测量。

问题如下：

排列具有 $N$ 个原子的分子，使两两原子之间的距离符和实验数据。

一个简单的例子如图 6-11 所示。

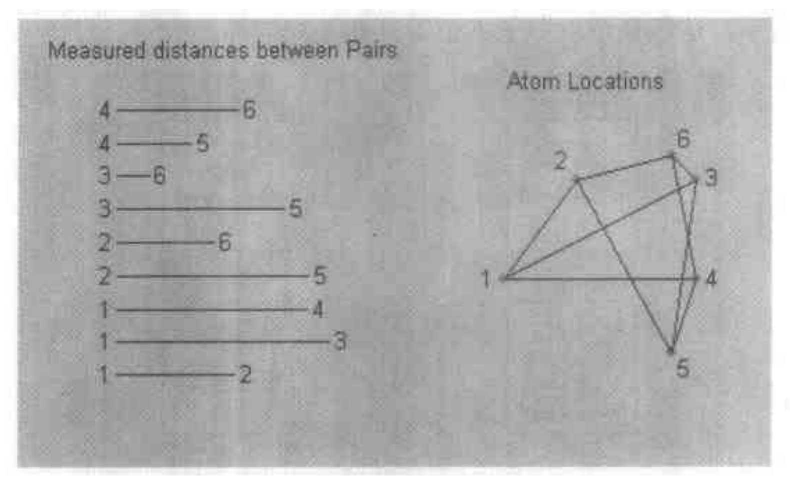



图 6-11 原子的实际位置

左边代表实验测量的实际距离，右边代表实际原子的位置。

2. 问题的分析

我们采用的方法是先把这些原子随机地分布成如图 6-12 所示，然后改变原子的位置，直到跟实验数据相吻合。令原子的位置表达式为：$X=\left[x(1:N)\,y(1:N)\right]'$，我们在测量数据中，令与每一对原子对应的误差为$\left[norm\left[X(i,:)-X(j,:)\right]\wedge 2-S(i,)\wedge 2\right]\wedge 2$，总的误差为 $MMOLE(X,S)$，是所有原子误差之和。我们的目的是找到使函数 $MMOLE(X,S)$最小的原子排列。

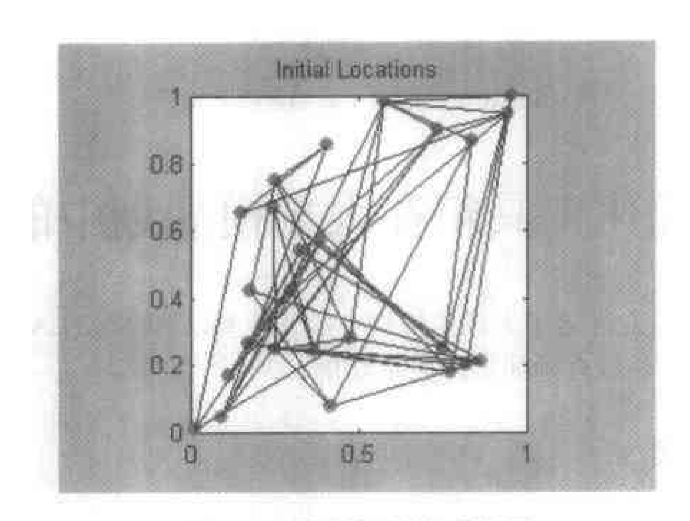



图 6-12 原子排列的初始位置

3. 问题的求解

由于位置的移动、旋转、反射并不改变$MMOLE(X,S)$的值。为了避免这些行为，我们令其中的三个原子是不动的，既然它们的位置已知，那么我们只有（2N-6）个未知量。

如图 6-13 所示，这四幅图中原子的位置其实是相同的。

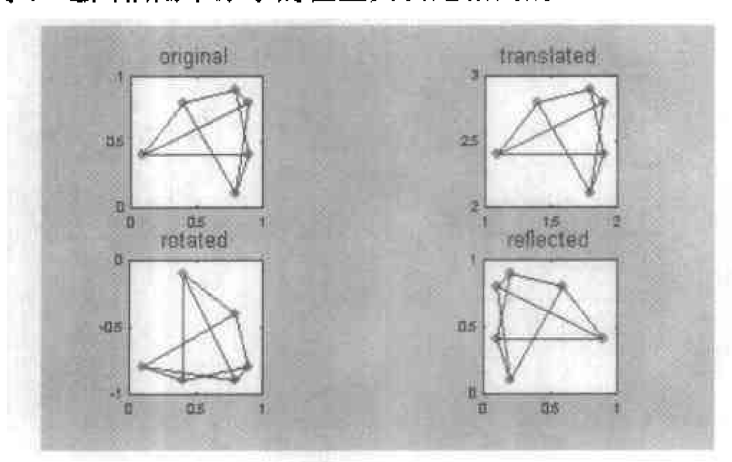



图 6-13 位置的移动、旋转、反射

让我们装入有 25 个原子的例子。S 是一个 25×25 稀疏矩阵，矩阵中每一个非零元素代表一个已知距离。我们可用命令来观察它的稀疏结构。

```
load moledata;
spy(S+S')
```

图 6-14 所示为 S 矩阵的结构。

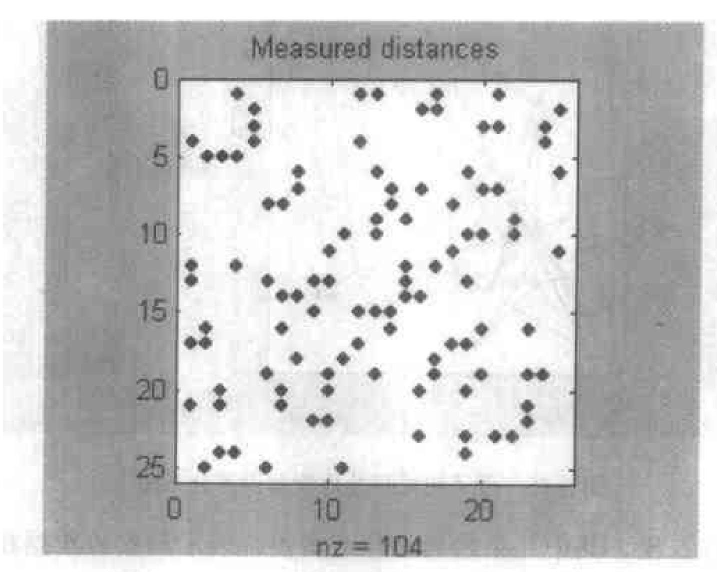



图 6-14　*S* 矩阵的结构

MMOLE 为定义的一个目标函数，它以现有原子的位置和距离稀疏矩阵为参数，计算出当前的距离矩阵、梯度函数、稀疏海色矩阵。

```
[val,g,H]=mmole(x,S);
spy(H)
```

其输出如图 6-15 所示。

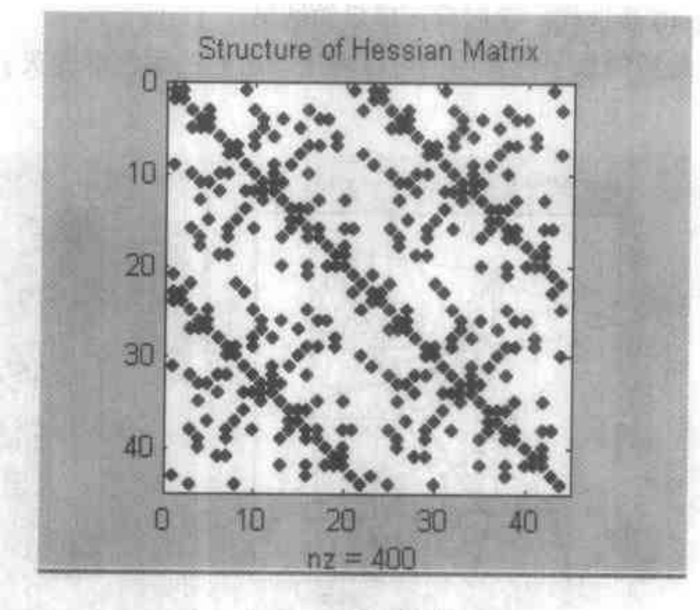



图 6-15　海色矩阵 *H*

因为 $H$ 是稀疏矩阵，我们可以用大规模算法来解决这个问题。

如图 6-16 所示，这是原子随机起始位置图。此位置与实验数据的误差用不同的颜色表示，以示误差的大小。

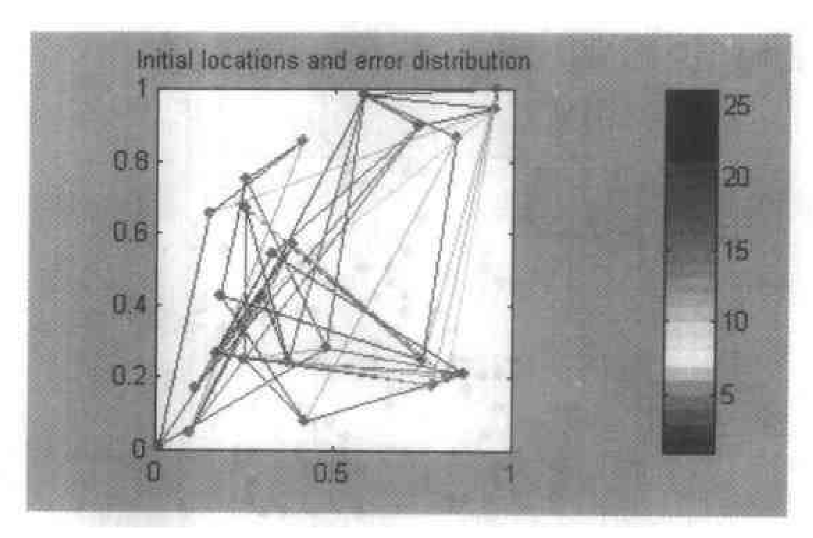



图 6-16 原子随机起始位置和误差分布图

由图 6-16 可以看出，其中几条线为红色或黄色，表示与实验数据相差很大。我们的目的就是减小这些误差。

我们现在用 options 命令为优化函数 FMINUNC 设置参数。我们采用在函数中计算的梯度和海色矩阵信息，产生优化的过程统计数据和迭代信息。

```
options = optimset('largescale','on','display','none','showstatus','iterplus', 'gradobj',"on',
'hessian','on');
```

现在使用函数得到优化结果：

```
x = fminunc('mmole',xstart,options,S);
```

可以看出，经过 40 步迭代，我们得到最后的结果。

图 6-17 所示是最后的原子位置图。可以看出，所有的颜色都变灰了，意味着与实验数据非常符合。

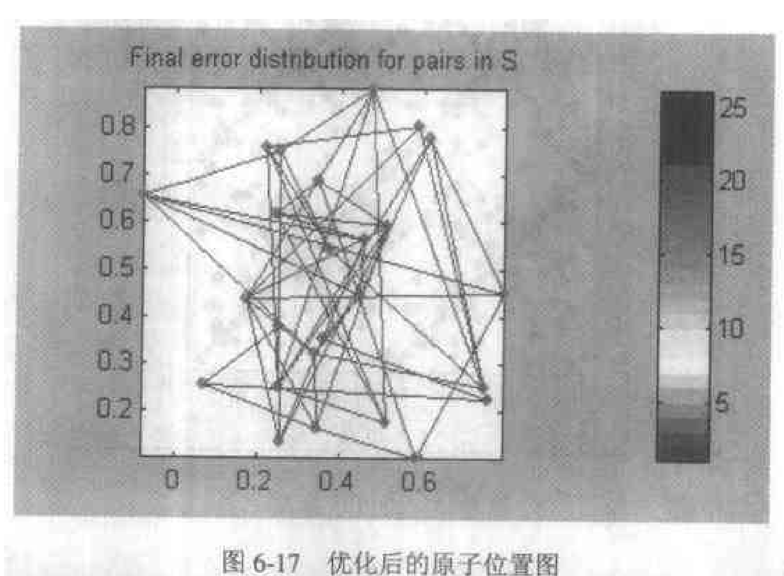



图 6-17 优化后的原子位置图

## 6.8.2　马戏团帐篷曲面形成问题

### 1．问题的提出

马戏团的帐篷形状问题实际上是一个有约束的优化问题。

假设马戏团的帐篷要覆盖一块长方形的地方。此帐篷由 5 个柱子支撑，上面覆盖一块布形成（如图 6-18 所示）。我们的目的是找到这个帐篷的天然形成的形状。这个形状对应于一个函数的极小值，而这个函数与表面的位置和它的位置梯度的平方误差有关。

### 2．问题的分析

这 5 个柱子决定了帐篷的下限值。

```
L = zeros(30);
E = ones(2);
L(15:16,15:16) = .5*E;
L(5:6,5:6) = .3*E;
L(25:26,5:6) = .3*E;
L(5:6,25:26) = .3*E;
L(25:26,25:26) = .3*E;
```

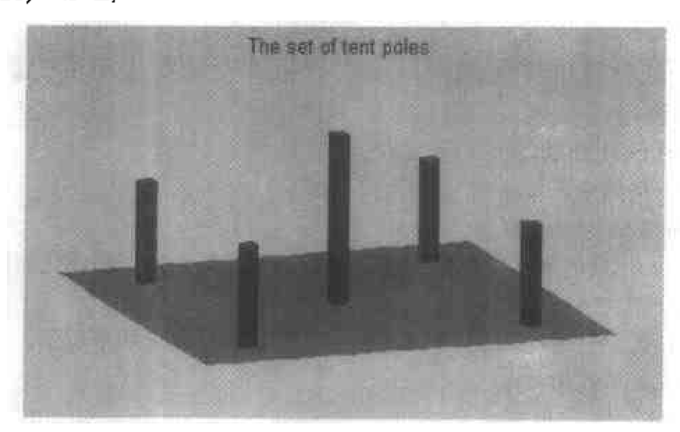



图 6-18　帐篷支撑柱的位置

我们可以画出帐篷中每个位置受约束的网格图形，如图 6-19 所示。

```
mesh(L,'edgecolor','m')
```

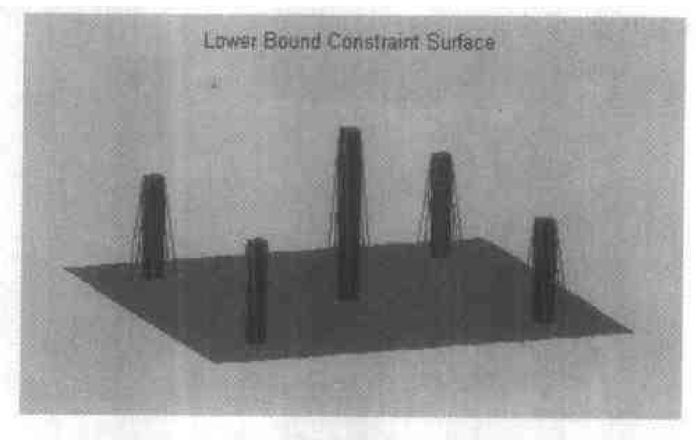



图 6-19　受约束的网格图形

为解决这个问题，我们首先假设帐篷由一系列的点组成。我们的目标就转化为找出所有点的高度。我们假设初始状态时，所有点的高度如图 6-20 所示。

```
sstart=.5*ones(30,30);
s= surf(sstart)
set(s,'FaceColor','none','LineStyle','none','Marker', '.', 'MarkerEdgeColor','b');'
```

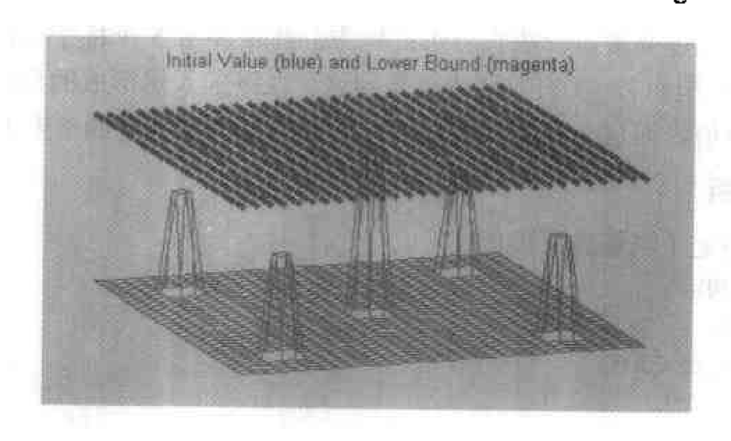



图 6-20　帐篷的初始位置

3. **问题的求解**

为了把此问题变成标准的优化问题，我们把矩阵变量变成向量变量。

```
%L 代表初始值。
    low = reshape(L,900,1);
%xstart 代表下端约束。
    xstart=reshape(sstart,900,1);
%上面覆盖的布形成的表面，也即下面函数所表示的线性约束极小化问题。
min{ c'*x+0.5*x'*H*x : low <= x }
%其中，c''*x+0.5*x''*H*x 是目标函数的离散逼近，H 和 c 如下所示。
H = delsq(numgrid('S',30+2));
h = 1/30-1;
c = -h^2*ones(30,1);
%目标函数的每一处的值仅与它的邻域有关。通常它的海色矩阵是稀疏的，有特殊的结构。
%所以我们可以用大规模算法来解决这个问题。
spy(H);
%输出如图 6-21 所示，海色矩阵是稀疏对角矩阵。
```

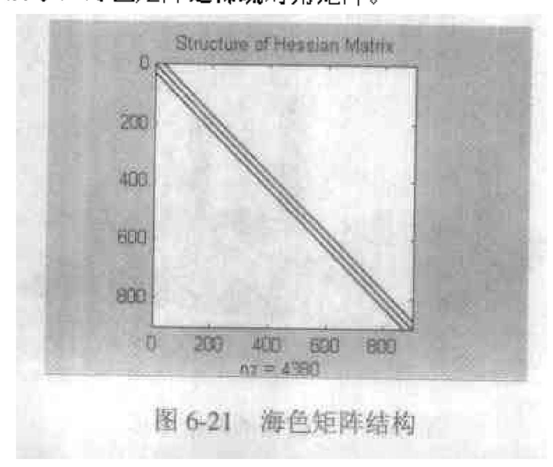



图 6-21　海色矩阵结构

```
%我们在 Option 中设置优化参数。
options = optimset('LargeScale','on', 'Display', 'off',    'ShowStatusWindow', 'iterplus');
%现在采用优化函数 QUADPROG 来解这个问题
x = quadprog(H, c, [], [], [], [], low, [], xstart, options);
%取得最优结果后，现在恢复原来的矩阵形式。
S = reshape(x,30,30);
%绘出最后的结果，如图 6-22 所示：
mesh(S,'edgecolor','b','facecolor','none');
```

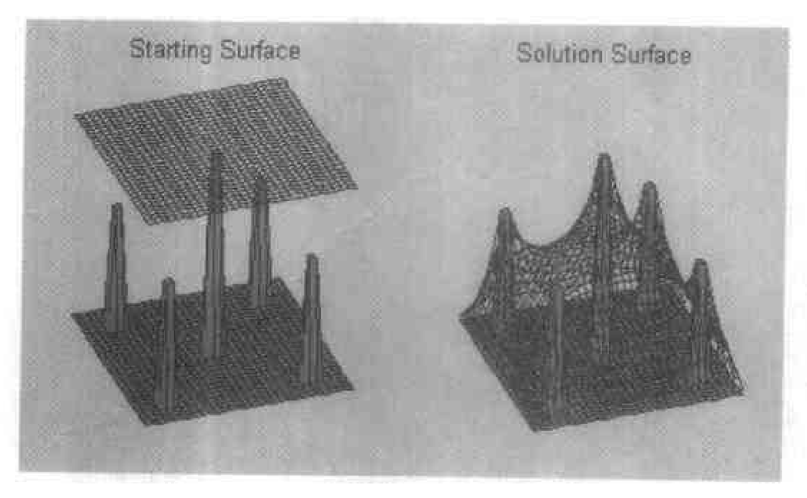



图 6-22　优化前后的结果对比

```
%图 6-23 是最后形成的马戏团帐篷的形状。
surf(L,'facecolor',[0 0 0]);
```

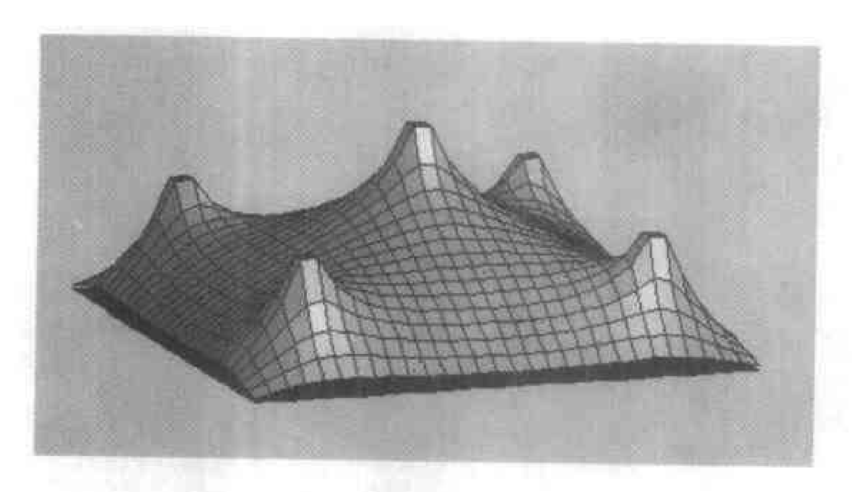


图 6-23　最后形成的马戏团帐篷的形状



# 附录 A　MATLAB 命令和函数参考

由于前面章节主要讲述了 MATLAB 优化工具箱中函数的用法，所以对于 MATLAB 标准环境下其他的函数和命令没有介绍，为了内容的完整和查询方便起见，现将这些命令和函数在附录中单独给出。

MATLAB 系统提供近 20 类基本命令函数，它们有一部分是 MATLAB 的内部命令，另一部分是以.m 文件形式给出的函数，这些.m 文件按照各自的类别归于相应的子目录下，每一个函数文件都包含了这一函数的用法指南，可以用命令：

```
help function_name
```

来显示函数 function_name 的帮助信息，也可以用命令：

```
help dictionary_name
```

来显示目录 dictionary_name 下一个函数文件的简要说明。

本附录仅给出各个函数的功能描述，用户可以用 help 命令获得相应函数的联机帮助，从而得到详细的函数信息（以下表中函数或命令名称按字母顺序排列）。

## A.1　常用命令参考

常用命令列出如表 A-1～表 A-4 所示。

表 A-1　运算符和特殊字符

| 符号名称 | 功能描述 |
| --- | --- |
| + | 加 |
| - | 减 |
| * | 矩阵相乘 |
| .* | 向量相乘 |
| ^ | 矩阵的幂运算 |
| .^ | 向量的幂运算 |
| \ | 矩阵左除 |
| / | 矩阵右除 |
| .\ | 向量左除 |
| ./ | 向量右除 |
| : | 向量生成或子阵提取 |
| () | 下标或参数定义 |
| [] | 矩阵生成 |



（续表）

| 符号名称 | 功能描述 |
| --- | --- |
| . | 结构字段获取符 |
| .. | 父目录 |
| ... | 继续标志 |
| , | 分行符（该行结果显示） |
| ; | 分行符（该行结果不显示） |
| % | 注释标志 |
| ! | 命令提示符 |
| ' | 矩阵转置 |
| .' | 向量转置 |
| = | 赋值符 |
| == | 关系运算符：相等 |
| <> | 关系运算符：不相等 |
| < | 关系运算符：小于 |
| <= | 关系运算符：不大于 |
| > | 关系运算符：大于 |
| >= | 关系运算符：不小于 |
| & | 逻辑运算符：与 |
| \| | 逻辑运算符：或 |
| ~ | 逻辑运算符：非 |
| xor | 逻辑运算符：异或 |
| kron | 矩阵 kron 积 |

**表 A-2　管理命令**

| 符号名称 | 功能描述 |
| --- | --- |
| addpath | 增加一条搜索路径 |
| clear | 从内存中删除变量和函数 |
| demo | 运行 MATLAB 演示程序 |
| disp | 显示矩阵或文本 |
| doc | 装入超文本说明 |
| help | 启动联机帮助 |
| lasterr | 显示最后一条错误信息 |
| length | 向量的维数 |
| load | 从文件中装入数据 |
| lookfor | 关键词搜索帮助 |
| pack | 整理工作空间内存 |
| path | 设置 MATLAB 路径 |

（续表）

| 符号名称 | 功能描述 |
|---|---|
| rmpath | 删除一条搜索路径 |
| save | 保存工作空间变量 |
| size | 矩阵的维数 |
| type | 列出.m 文件 |
| version | 显示 MATLAB 版本 |
| what | 列出当前目录下的.m，.mex，.mat 文件 |
| whatsnew | 显示新特性 |
| which | 定位函数或文件的目录 |
| who,whos | 列出 MATLAB 工作空间的变量 |

表 A-3 文件和操作系统命令

| 符号名称 | 功能描述 |
|---|---|
| cd | 改变当前工作目录 |
| delete | 删除文件 |
| diary | 保存 MATLAB 运行命令 |
| edit | 编辑.m 文件 |
| ! | 执行操作系统命令 |
| getenv | 获取环境变量值 |
| matlabroot | 获取 MATLAB 安装跟目录 |
| tempdir | 获取系统的缓存目录 |
| tempname | 获取一个缓存文件 |
| unix | 执行操作系统命令并返回结果 |

表 A-4 窗口控制命令

| 符号名称 | 功能描述 |
|---|---|
| cedit | 命令行编辑 |
| clc | 清除命令窗口 |
| echo | 显示文件中使用的 MATLAB 的命令 |
| format | 设置输出格式 |
| home | 设置光标位于左上角 |
| hostid | MATLAB 主服务程序的代号 |
| info | MATLAB 系统信息 |
| matlabbrc | 启动主.m 文件 |
| more | 控制命令窗口分页输出 |
| quit | 退出 MATLAB |
| startup | MATLAB 自启动文件 |
| subscribe | 定购 MATLAB |





# A.2 常用函数参考

常用函数列出如表 A-5～表 A-24 所示。

表 A-5 逻辑函数

| 函数名称 | 功能描述 |
| --- | --- |
| All | 向量的所有元素为真时，返回 true |
| Any | 向量的任一元素为真时，返回 true |
| Exist | 检验变量或函数是否存在 |
| Find | 查找非零元素的索引号 |
| Finite | 含有有限元时，返回 true |
| Isempt | 矩阵为空时，返回 true |
| Isglobal | 变量为全局变量时，返回 true |
| Isinf | 含有无限大元时，返回 true |
| Isnan | 含有 NaN 时，返回 true |
| Isreal | 矩阵为实矩阵时，返回 true |
| Issparse | 矩阵为稀疏矩阵时，返回 true |
| Isstr | 矩阵为字符串时，返回 true |
| Logical | 将数字量转化为逻辑量 |

表 A-6 数学函数之一：三角函数

| 函数名称 | 功能描述 |
| --- | --- |
| acos | 反余弦函数 |
| acosh | 反双曲余弦函数 |
| acot | 反余切函数 |
| acoth | 反双曲余切函数 |
| acsc | 反余割函数 |
| acsch | 反双曲余割函数 |
| asin | 反正弦函数 |
| asinh | 反双曲正弦函数 |
| atan | 反正切函数 |
| atan2 | 反双曲正切函数 |
| cos | 余弦函数 |
| cosh | 双曲余弦函数 |
| cot | 余切函数 |
| coth | 双曲余切函数 |

（续表）

| 函数名称 | 功能描述 |
| --- | --- |
| csc | 余割函数 |
| csch | 双曲余割函数 |
| sec | 正割函数 |
| sech | 双曲正割函数 |
| sin | 正弦函数 |
| sinh | 双曲正弦函数 |
| tan | 正切函数 |
| tanh | 双曲正切函数 |

表 A-7　数学函数之二：指数函数

| 函数名称 | 功能描述 |
| --- | --- |
| exp | 指数函数 |
| log | 自然对数函数 |
| log10 | 常用对数函数 |
| sqrt | 平方根函数 |

表 A-8　数学函数之三：复数函数

| 函数名称 | 功能描述 |
| --- | --- |
| abs | 绝对值函数 |
| angle | 相角函数 |
| conj | 共轭复数函数 |
| imag | 复数虚部函数 |
| real | 复数实部函数 |

表 A-9　数学函数之四：数值处理函数

| 函数名称 | 功能描述 |
| --- | --- |
| ceil | 沿正无穷方向取整函数 |
| fix | 沿零方向取整函数 |
| floor | 沿负无穷方向取整函数 |
| rem | 求余函数 |
| round | 取整函数 |
| sign | 符号函数 |

表 A-10　数学函数之五：特殊数学函数

| 函数名称 | 功能描述 |
| --- | --- |
| airy | / |

（续表）

| 函数名称 | 功能描述 |
| --- | --- |
| besselh | Bessel 函数 |
| besseli | 改进的第一类 Bessel 函数 |
| besselj | 第一类 Bessel 函数 |
| besselk | 改进的第二类 Bessel 函数 |
| bessely | 第二类 Bessel 函数 |
| beta | β函数 |
| betainc | 非完全的β函数 |
| betaln | β函数的对数函数 |
| cart2pol | 变笛卡儿坐标为极坐标函数 |
| cart2sph | 变笛卡儿坐标为球坐标函数 |
| ellpj | Jacobian 椭球函数 |
| ellpke | 椭圆积分函数 |
| erf | 误差函数 |
| erfc | 互补误差函数 |
| erfcx | 比例互补误差函数 |
| erfinv | 逆误差函数 |
| expint | 指数积分函数 |
| gamma | γ函数 |
| gammainc | 非完全γ函数 |
| gammaln | γ函数对数函数 |
| gcd | 最大公约数 |
| lcm | 最小公倍数 |
| log2 | 分割浮点数 |
| legendre | Legendre 伴随函数 |
| pol2cart | 变极坐标为笛卡儿坐标函数 |
| pow2 | 比例浮点函数 |
| sph2cart | 变球坐标为笛卡儿坐标函数 |
| rat | 有理逼近 |
| rats | 有理输出 |

表 A-11　数值分析函数之一：多项式处理函数

| 函数名称 | 功能描述 |
| --- | --- |
| conv | 多项式乘法 |
| deconv | 多项式除法 |
| poly | 构造具有特定根的多项式 |

（续表）

| 函数名称 | 功能描述 |
| --- | --- |
| polyder | 多项式求微分 |
| polyeig | 求多项式特征值 |
| polyfit | 数据的多项式拟合 |
| polyval | 计算多项式的值 |
| polyvalm | 计算多项式矩阵 |
| residue | 留数计算 |
| roots | 计算多项式的根 |

表 A-12　数值分析函数之二：插值函数

| 函数名称 | 功能描述 |
| --- | --- |
| griddata | 生成数据网格 |
| interp1 | 一维插值 |
| interp2 | 二维插值 |
| interp3 | 三维插值 |
| interpft | 利用 FFT 进行一维插值 |
| interpn | 多维插值 |
| meshgrid | 用 $x$，$y$，$z$ 构造三维图形 |
| ppval | 分段多项式计算 |
| spline | 三次样条插值 |

表 A-13　数值分析函数之三：非线性数值计算函数

| 函数名称 | 功能描述 |
| --- | --- |
| dblquad | 双重积分 |
| fmin | 单变量最优化函数 |
| fmins | 多变量最优化函数 |
| fzero | 求解单变量函数的零点 |
| ode23 | 低阶法求解常微分方程 |
| ode23p | 低阶法求解常微分方程并绘图 |
| ode45 | 高阶法求解常微分方程 |
| odefile | 求解由文件定义的微分方程 |
| odeget | 获得微分方程求解的可选参数 |
| odeset | 设置微分方程求解的可选参数 |
| quad | 低阶法计算数值积分 |
| quad8 | 高阶法计算数值积分 |

表 A-14　数据分析函数

| 函数名称 | 功能描述 |
| --- | --- |
| conv | 卷积 |
| conv2 | 二维卷积 |
| corrcoef | 计算相关系数 |
| cov | 计算协方差矩阵 |
| cross | 向量的矢量积 |
| cumprod | 向量累积 |
| cumsum | 向量累加 |
| deconv | 反卷积 |
| del2 | 离散拉普拉斯变换 |
| dot | 向量点积 |
| diff | 计算差分或近似微分 |
| filter | 一维数字滤波 |
| filter2 | 二维数字滤波 |
| gradient | 计算梯度 |
| max | 计算向量中的最大分量 |
| mean | 计算向量中的各分量均值 |
| median | 计算向量中的中间分量 |
| min | 计算向量中的最小分量 |
| prod | 计算各分量之积 |
| sort | 对分量按升序排列 |
| sortrows | 对矩阵各行排序 |
| std | 计算向量各元素的标准差 |
| sum | 计算向量中各分量之和 |
| trapz | 利用梯形法计算数值积分 |

表 A-15　傅里叶变换函数

| 函数名称 | 功能描述 |
| --- | --- |
| abs | 绝对值函数 |
| angle | 相角函数 |
| cplxpair | 按共轭复数对进行排序 |
| fft | 离散傅里叶变换 |
| fft2 | 二维离散傅里叶变换 |
| fftshift | 将零点平移到频谱中心 |
| nextpow2 | 最靠近 2 的幂次数 |
| ifft | 离散傅里叶逆变换 |

（续表）

| 函数名称 | 功能描述 |
|---|---|
| ifft2 | 二维离散傅里叶逆变换 |
| unwrap | 相角矫正 |

表 A-16 矩阵函数之一：基本矩阵函数

| 函数名称 | 功能描述 |
|---|---|
| ans | 默认的计算结果变量 |
| computer | 当前的计算机型号 |
| eps | 浮点精度 |
| eye | 生成单位矩阵 |
| flops | 浮点运算次数 |
| i | 虚数单元 |
| Inf | 无穷大 |
| inputname | 输入参数名 |
| isieee | 当采用 IEEE 算术标准时，返回 true |
| j | 虚数单元 |
| linspace | 构造线性分布的向量 |
| logspace | 构造对数分布的向量 |
| nargin | 函数输入变量的数目 |
| nargout | 函数输出变量的数目 |
| NaN | 非数值常量 |
| ones | 生成所有元素为“1”的矩阵 |
| pi | 圆周率 |
| rand | 产生随机分布矩阵 |
| randn | 产生正态分布矩阵 |
| realmax | 最大浮点数 |
| realmin | 最小浮点数 |
| varargin | 函数中输入的可选参数 |
| varargout | 函数中输出的可选参数 |
| zeros | 产生零矩阵 |

表 A-17 矩阵函数之二：稀疏矩阵函数

| 函数名称 | 功能描述 |
|---|---|
| colmmd | 列最小度排序 |
| colperm | 按非零元素的个数来排列各列 |
| condest | 范数估计 |
| dmperm | Dulmage-Mendelsohn 分解 |

（续表）

| 函数名称 | 功能描述 |
| --- | --- |
| eigs | 求稀疏矩阵的特征值和特征向量 |
| etree | 求矩阵的消元树 |
| etreeplot | 画消元树图 |
| find | 查找非零元素的序号 |
| full | 变稀疏矩阵为常规矩阵 |
| issparse | 判断是否为稀疏矩阵 |
| gplot | 绘图 |
| nnz | 稀疏矩阵非零元素的个数 |
| nonzeros | 稀疏矩阵的非零元素 |
| nzmax | 允许的非零元素空间 |
| randperm | 产生随机置换向量 |
| spalloc | 为非零元素定位存储空间 |
| sparse | 变常规矩阵为稀疏矩阵 |
| spaugment | 形成最小二乘增广系统 |
| spconvert | 由外部格式引入稀疏矩阵 |
| spdiags | 稀疏对角矩阵 |
| speye | 稀疏单位矩阵 |
| spfun | 为非零元素定义处理函数 |
| spones | 将零元素替换为 1 |
| spparms | 设置稀疏矩阵参数 |
| sprand | 稀疏均匀分布随机矩阵 |
| sprandn | 稀疏正态分布随机矩阵 |
| sprandsym | 稀疏对称随机矩阵 |
| spy | 绘制稀疏矩阵结构 |
| svds | 稀疏矩阵奇异值分解 |
| symbact | 符号因子分解 |
| symmd | 最小对称度 |
| symrcm | 逆 Cathill-Mckee 序 |

**表 A-18　矩阵函数之三：特殊矩阵函数**

| 函数名称 | 功能描述 |
| --- | --- |
| compan | 伴随矩阵 |
| gallery | 生成一些小的测试矩阵 |
| hadamard | 生成 Hadamard 矩阵 |
| hankel | 生成 Hankel 矩阵 |
| hilb | 生成 Hilbert 矩阵 |

（续表）

| 函数名称 | 功能描述 |
|---|---|
| invhilb | 逆 Hilbert 矩阵 |
| kron | Kronecker 张量积 |
| magic | 生成魔方矩阵 |
| pascal | 生成 PASCAL 矩阵 |
| rosser | 经典的对称特征值测试问题 |
| toeplitz | 生成 Toeplitz 矩阵 |
| Vander | 生成 Vandermonde 矩阵 |
| wilkinson | 生成 Wilkinson 特征值测试矩阵 |

**表 A-19　矩阵函数之四：矩阵分析与处理函数**

| 函数名称 | 功能描述 |
|---|---|
| balance | 矩阵均衡处理 |
| cat | 向量连接 |
| cdf2rdf | 变复块对角矩阵为实块对角矩阵 |
| cond | 计算矩阵条件数 |
| diag | 提取对角阵 |
| eig | 计算矩阵特征值和特征向量 |
| expm | 矩阵指数函数 |
| expm1 | 实现 expm 的.m 文件 |
| expm2 | 通过泰勒级数计算矩阵指数 |
| expm3 | 通过特征值和特征向量计算矩阵指数 |
| fliplr | 矩阵做左右翻转 |
| flipud | 矩阵做上下翻转 |
| funm | 一般矩阵计算函数 |
| det | 计算矩阵行列式 |
| hess | 计算 Hessberg 矩阵 |
| logm | 矩阵对数函数 |
| norm | 计算矩阵范数 |
| null | 零空间 |
| orth | 正交化函数 |
| poly | 计算特征多项式 |
| polyeig | 多项式特征值问题 |
| qz | 计算广义特征值 |
| rank | 计算矩阵的秩 |
| rcond | LINPACK 倒数条件估计 |
| repmat | 复制并排列矩阵函数 |





（续表）

| 函数名称 | 功能描述 |
| --- | --- |
| reshape | 改变矩阵大小 |
| rot90 | 矩阵旋转 90° |
| rref | 矩阵的行阶梯形实现 |
| rrefmovie | 消元法解方程演示 |
| rsf2csf | 变实块对角矩阵为复块对角矩阵 |
| schur | Schur 分解 |
| sqrtm | 计算矩阵平方根函数 |
| subspace | 子空间 |
| svd | 奇异值分解函数 |
| trace | 计算矩阵的迹 |
| tril | 提取矩阵下三角部分 |
| triu | 提取矩阵上三角部分 |

表 A-20　矩阵函数之五：线性方程分析函数

| 函数名称 | 功能描述 |
| --- | --- |
| \和/ | 求解线性方程 |
| chol | Cholesky 分解 |
| inv | 矩阵求逆 |
| lscov | 根据协方差计算最小二乘方差 |
| lu | 矩阵的 LU 三角分解 |
| nnls | 非零最小二乘 |
| pinv | 计算伪逆矩阵 |
| qr | 矩阵的 QR 分解 |
| qrdelete | 从 QR 分解中消除一行 |
| qrinsert | 在 QR 分解中插入一行 |

表 A-21　图形函数之一：通用图形控制函数

| 函数名称 | 功能描述 |
| --- | --- |
| axes | 建立坐标系 |
| axis | 设置坐标系标度 |
| box | 设置坐标系为盒状 |
| capture | 抓取屏幕当前图形 |
| caxis | 设置彩色坐标轴刻度 |
| cla | 清除当前坐标系 |
| clf | 清除当前图形 |
| close | 关闭图形窗口 |

（续表）

| 函数名称 | 功能描述 |
|---|---|
| copyobj | 拷贝图形对象 |
| cylinder | 生成圆柱体 |
| delete | 删除图形对象 |
| drawnow | 清除未完成的绘图事件 |
| figure | 建立图形窗口 |
| findobj | 查找指定对象 |
| gca | 获得当前坐标轴句柄 |
| gcf | 获得当前图形的窗口句柄 |
| gco | 获得当前对象的句柄 |
| gcbf | 获得当前回调窗口的句柄 |
| get | 获得对象属性 |
| getframe | 获得动画帧 |
| ginput | 用鼠标输入图形 |
| graymon | 设置图形为灰色显示器的默认值 |
| hold | 保持当前图形 |
| ishold | 返回 hold 状态 |
| light | 生成光源 |
| line | 生成直线 |
| movie | 播放记录的动画帧 |
| moviein | 初始化动画帧内存 |
| orient | 设置纸张纸向 |
| patch | 建立图形填充块 |
| rbbox | 建立涂抹块 |
| refresh | 刷新图形窗口 |
| reset | 重新设置对象属性 |
| rotate | 沿指定方向旋转对象 |
| set | 设置对象属性 |
| shg | 显示图形窗口 |
| sphere | 生成球 |
| subplot | 将图形窗口分区 |
| surface | 建立曲面 |
| terminal | 设置图形终端类型 |
| text | 生成文本串 |
| unicontrol | 生成一个用户接口控制 |
| uimenu | 生成菜单 |
| waitforbuttonpress | 在图形窗口等待按键 |

（续表）

| 函数名称 | 功能描述 |
| --- | --- |
| whitebg | 设置图形窗口为白色背景的默认值 |
| zoom | 图形的缩放 |

表 A-22　图形函数之二：二维图形函数

| 函数名称 | 功能描述 |
| --- | --- |
| area | 区域填充 |
| bar | 绘制条形图 |
| barh | 绘制水平条形图 |
| comet | 绘制彗星状轨迹 |
| compass | 绘制区域图 |
| errbar | 绘制误差条形图 |
| feather | 绘制羽状图形 |
| fill | 绘制二维多边形填充图 |
| fplot | 绘制给定函数 |
| grid | 生成网格线 |
| hist | 生成直方图 |
| loglog | 绘制对数坐标图形 |
| pareto | 绘制 Pareto |
| pie | 绘制饼状图 |
| plot | 绘制线性坐标图形 |
| polar | 绘制极坐标图形 |
| rose | 绘制角度直方图 |
| semilogx | 绘制 $x$ 轴半对数坐标图形 |
| semilogy | 绘制 $y$ 轴半对数坐标图形 |
| stairs | 绘制阶梯图 |
| stem | 绘制离散序列图形 |
| title | 绘制图形标题 |
| xlabel | 绘制 $x$ 轴标记 |
| ylabel | 绘制 $y$ 轴标记 |
| zlabel | 绘制 $z$ 轴标记 |

表 A-23　图形函数之三：三维图形函数

| 函数名称 | 功能描述 |
| --- | --- |
| bar3 | 绘制三维条形图形 |
| bar3h | 绘制三维水平条形图形 |
| brighten | 加亮图形色调 |

（续表）

| 函数名称 | 功能描述 |
|---|---|
| caxis | 设置坐标轴伪彩色 |
| colormap | 设置调色板 |
| comet3 | 绘制三维彗星状轨迹 |
| contour | 绘制等高线 |
| contourf | 绘制填充的等高线 |
| contour3 | 绘制三维等高线 |
| clabel | 等高线高程标志 |
| fill3 | 绘制并填充三维多边形 |
| hidden | 设置网格图的网格线开关 |
| mesh | 绘制三维网格图形 |
| meshc | 绘制网格和等高线的混合图形 |
| meshz | 绘制带零平面的三维网格图 |
| pcolor | 绘制伪彩色 |
| plot3 | 绘制三维线或点型图形 |
| quiver | 绘制有向图 |
| quiver3 | 绘制三维有向图 |
| rotate3d | 设置三维旋转开关 |
| shading | 设置彩色阴影 |
| slice | 绘制切片图 |
| stem3 | 绘制三维杆图 |
| surf | 绘制三维表面图形 |
| surfc | 绘制三维网格和等高线的混合图形 |
| surfl | 绘制带亮度的三维曲面图 |
| surfnorm | 绘制曲面法线 |
| trisurf | 表面图形的三角绘制 |
| trimesh | 网格图形的三角绘制 |
| view | 设置视点 |
| voronoi | 绘制 Voronoi |
| waterfall | 绘制瀑布型图形 |

**表 A-24　图形函数之四：用户图形接口函数**

| 函数名称 | 功能描述 |
|---|---|
| align | 坐标轴对齐工具 |
| axlimdlg | 坐标轴范围设置对话框 |
| btndown | 按下组按钮中的按钮 |

（续表）

| 函数名称 | 功能描述 |
|---|---|
| btngroup | 生成组按钮 |
| btnpress | 按下管理组按钮中的按钮 |
| btnstate | 查询组按钮中的按钮状态 |
| btnup | 组按钮中的按钮弹起 |
| cbedit | 回调函数编辑器 |
| dialog | 主对话框的.m 文件 |
| errordlg | 错误对话框 |
| guide | GUI 设计工具 |
| helpdlg | 帮助对话框 |
| inputdlg | 输入对话框 |
| layout | 定义对话框布局参数 |
| listdlg | 列表选择对话框 |
| makemenu | 生成菜单结构 |
| menubar | 设置菜单条属性 |
| menuedit | 菜单编辑器 |
| msgdlg | 消息对话框 |
| pagedlg | 页位置对话框 |
| printdlg | 打印对话框 |
| propedit | 属性编辑器 |
| prtps | PostScript 打印机驱动程序 |
| prtwin | MS Windows 驱动程序 |
| questdlg | 请求对话框 |
| uicontrol | 生成图形用户接口对象 |
| uimenu | 生成菜单对象 |
| uigetfile | 标准的打开文件对话框 |
| uiputfile | 标准的保存文件对话框 |
| uiresume | 继续执行 |
| uisetcolor | 颜色选择对话框 |
| uisetfont | 字体选择对话框 |
| uiwait | 中断执行 |
| umtoggle | 切换选中的菜单对象 |
| waitbar | 等待条显示 |
| waitfor | 中断执行 |
| waitferbutterpress | 等待按钮事件 |
| warndlg | 警告对话框 |
| wimenu | 生成菜单项的子菜单 |



# A.3　工具箱函数参考

MATLAB 系统的主要特点就是它本身提供很多工具箱（toolbox），用户可以调用其中的大量函数。目前 MATLAB 的工具箱有 30 多个，而且由于其自身的可扩展性，从而由第三方开发的工具箱正日益增多。本附录列出了其中一部分常用工具箱提供的函数，如表 A-25～表 A-34 所示。

表 A-25　控制系统工具箱（Control System Toolbox）

| 类　别 | 函数名称 | 功能描述 |
|---|---|---|
| 模型的特性 | covar | 输入为白噪声的连续协方差响应 |
| | ctrb | 可控矩阵 |
| | damp | 连续阻尼系数和固有频率 |
| | dcgain | 连续稳态增益 |
| | dcovar | 输入为白噪声的离散协方差响应 |
| | ddamp | 离散阻尼系数和固有频率 |
| | ddcgain | 离散稳态增益 |
| | dsort | 按幅值大小排列离散特征值 |
| | eig | 特征值和特征向量 |
| | esort | 按实部大小排列连续特征值 |
| | gram | 可控性和可观性 |
| | obsv | 可观性矩阵 |
| | printsys | 设置显示格式 |
| | roots | 多项式的根 |
| | tzero | LTI 系统的传递零点 |
| | tzero2 | 传递零点 |
| | | |
| 简化模型 | balreal | 基于 Giamian 平衡的状态空间实现 |
| | dbalreal | 离散平衡状态空间的实现和模型简化 |
| | dmodred | 离散时间模型状态降阶 |
| | minreal | 最小实现和零极点相消 |
| | modred | 模型降阶 |
| | | |
| 模型实现 | canon | 正则状态空间实现 |
| | ctrbf | 可控阶梯形 |
| | obsvf | 可观阶梯形 |
| | ss2ss | 状态空间模型的相似变换 |

（续表）

| 类　别 | 函数名称 | 功能描述 |
|---|---|---|
| 建立模型 | append | 增加系统动态特性 |
| | augstate | 以变量状态作为状态空间的输出 |
| | blkbuild | 由传递函数方框图构造状态空间结构 |
| | cloop | 闭环系统 |
| | connect | 由方框图构造状态空间模型 |
| | conv | 卷积 |
| | destim | 由增益矩阵构造离散状态估计器 |
| | dreg | 由增益矩阵构造离散控制器和估计器 |
| | drmodel | 生成离散随机模型 |
| | estim | 由增益矩阵构造连续状态估计器 |
| | feedback | 构造反馈系统 |
| | ord2 | 生成二阶系统的 A、B，C，D |
| | pade | Pade 的时延近似 |
| | parallel | 构造并行连接系统 |
| | reg | 由增益矩阵构造连续控制器和估计器 |
| | rmodel | 生成连续随机模型 |
| | series | 构造串行连接系统 |
| | ssdelete | 删除模型中的输入、输出状态 |
| | ssselect | 选择大系统中的子系统 |
| | | |
| 模型变换 | c2d | 变连续系统为离散系统 |
| | c2dm | 按指定方法变连续系统为离散系统 |
| | c2dt | 变连续系统为带一时延的离散系统 |
| | d2c | 变离散系统为连续系统 |
| | d2cm | 按指定方法变离散系统为连续系统 |
| | poly | 变根值表示为多项式表示 |
| | residue | 部分分式展开 |
| | ss2tf | 变状态空间表示为传递函数表示 |
| | ss2zp | 变状态空间表示为零极点表示 |
| | tf2ss | 变传递函数表示为状态空间表示 |
| | tf2zp | 变传递函数表示为零极点表示 |
| | zp2tf | 变零极点表示为传递函数表示 |
| | zp2ss | 变零极点表示为状态空间表示 |
| | | |

（续表）

| 类别 | 函数名称 | 功能描述 |
|---|---|---|
| 时域响应 | dimpulse | 离散单位冲击响应 |
| | dinitial | 离散零输入响应 |
| | dlsim | 任意输入的离散仿真 |
| | dstep | 离散阶跃响应 |
| | filter | 单输入单输出Z变换仿真 |
| | impulse | 冲击响应 |
| | initial | 零输入连续响应 |
| | lsim | 任意输入的连续仿真 |
| | step | 阶跃响应 |
| | stepfun | 阶跃函数 |
| | | |
| 频域响应 | bode | 波特图 |
| | dbode | 离散波特图 |
| | dnichols | 离散Nichols图 |
| | dnyquist | 离散Nyquist图 |
| | dsigma | 离散奇异值频域图 |
| | fbode | 连续系统波特图 |
| | freqz | Z变换频域响应 |
| | ltifr | 线性时变频率响应 |
| | margin | 增益和相位域度 |
| | nichols | Nichols图 |
| | ngrid | 画Nichols图的网格线 |
| | nyquist | Nyquist图 |
| | sigma | 奇异值频域图 |
| | | |
| 增益与根轨迹 | acker | 单输入单输出系统极点配置 |
| | dlqe | 离散线性二次估计器设计 |
| | dlqew | 带有过程噪声的离散系统Kalman估计器设计 |
| | dlqr | 离散线性二次调节器设计 |
| | dllqry | 输出加权的离散调节器设计 |
| | lqe | 线性二次估计器设计 |
| | lqed | 基于连续代价函数的离散估计器设计 |
| | lqe2 | 利用Schur方法设计线性二次估计器 |
| | lqew | 带有过程噪声的连续系统Kalman估计器设计 |
| | lqr | 线性二次调节器设计 |
| | lqrd | 基于连续代价函数的离散调节器设计 |

（续表）

| 类　别 | 函数名称 | 功能描述 |
|---|---|---|
| | lqry | 输出加权的调节器设计 |
| | place | 配置极点 |
| | pzmap | 零极点图 |
| | rlocfind | 给定根的轨迹 |
| | rlocus | 绘制根轨迹 |
| | sgrid | 生成根轨迹的 $s$ 平面网格 |
| | zgrid | 生成根轨迹的 $z$ 平面网格 |
| | | |
| 其他函数 | abcdchk | 检测（A，B，C，D）的一致性 |
| | boildemo | 锅炉系统的 LQG 设计演示示例 |
| | chop | 取出 $n$ 个重要的元素 |
| | ctrldemo | 控制工具箱介绍 |
| | dexresp | 离散取样响应函数 |
| | dfrqint | 离散波特图 |
| | dftqint2 | 离散波特图的自定范围算法 |
| | diskdemo | 硬盘控制器的数字控制演示示例 |
| | dmulresp | 离散多变量响应函数 |
| | dric | 计算 Riccati 离散方程留数 |
| | dsigma2 | Dsigma 实用工具函数 |
| | dtimvec | 离散响应的自定范围算法 |
| | exresp | 取样响应函数 |
| | freqint | 波特图的自定范围算法 |
| | freqint2 | Nyquist 图的自定范围算法 |
| | freqresp | LTI 模型的响应函数 |
| | givens | 旋转函数 |
| | jetdemo | 喷气式飞机偏航阻尼的设计演示示例 |
| | kalmdemo | Kalman 滤波器设计与仿真演示示例 |
| | housh | 构造 Housebuilder 变换 |
| | lab2ser | 变符号为字符串 |
| | mulresp | 多变量响应函数 |
| | nargchk | 检测变量 |
| | perpxy | 寻找最近的正交点 |
| | poly2str | 变多项式为字符串 |
| | printmat | 带行列号打印矩阵 |
| | ric | 计算 Riccati 方程留数 |

（续表）

| 类　别 | 函数名称 | 功能描述 |
|---|---|---|
| | schord | 有序 Schwr 分解 |
| | sigma2 | Sigma 实用工具函数 |
| | tfchk | 检测传递函数的一致性 |
| | timvec | 连续响应的自定范围算法 |
| | vsort | 匹配两轨迹的向量 |

表 A-26 非线性控制设计工具箱

| 类　别 | 函数名称 | 功能描述 |
|---|---|---|
| 用户界面工具 | coneddlg | 管理 NCD 编辑器的对话框 |
| | contrncd | 建立 NCD 固定图形的界面 |
| | curobj | 提供当前对象的信息 |
| | dividecb | 将固定界面分为两部分 |
| | delline | 删除 NCD 图中的所有图 |
| | donep | 收回 Close 按钮和菜单 |
| | errorncd | 管理 NCD 的出错信息 |
| | fillaxes | 建立约束边界并进行数据检测 |
| | forceit | 在已有的界限内插入一个子集 |
| | keyncd | NCD 键函数 |
| | loadncd | 载入并显示 NCD 数据 |
| | makesurf | 建立有边界约束的曲面 |
| | menuncd | 建立 NCD 固定图形的用户界面菜单 |
| | ncdblock | 包含 NCD 框图的 Simulink 系统 |
| | optblock | 打开一个 NCD 图形的底稿文件 |
| | optfig | 建立一个 NCD 块固定图形 |
| | paramdlg | 管理 NCD 优化参数的对话框 |
| | rangedlg | 管理坐标系范围的对话框 |
| | refdlg | 管理 NCD 参考信号的对话框 |
| | refresho | 保持约束矩阵和图形的一致性 |
| | snapncd | 以 22.5 栅格的间隔排出约束条 |
| | stepdlg | 管理 NCD 阶跃响应的对话框 |
| | texted | 收回 port 可编辑的文本 |
| | uncerdlg | 不确定变量帮助对话框 |
| | undatedlg | 更新 NCD 对话框 |
| | undoncd | 放弃上一步的 NCD 图形用户界面操作 |

（续表）

| 类 别 | 函数名称 | 功能描述 |
|---|---|---|
| NCD优化函数 | convertm | 变约束矩阵为最优化标准格式 |
| | costfun | NCD 优化的代价函数 |
| | minipars | NCD 最小化分析 |
| | montevar | Monte Carlo 仿真初始化 |
| | ncdglob | 定义 NCD 全局变量 |
| | nlinopt | 执行优化算法 |
| | str2mat2 | 变一行字符串为多行字符串 |
| | | |
| 演示及帮助函数 | hotkey | 热键帮助 |
| | mainncd | 一般 NCD 帮助 |
| | ncddemo | 包含所有 NCD 演示示例的 Simulink 系统 |
| | ncddemo1 | PID 控制器演示示例 |
| | ncddemo2 | 带前馈控制器的 LQR 演示示例 |
| | ncddemo3 | 多输入多输出的 PID 控制器演示示例 |
| | ncddemo4 | 倒摆演示示例 |
| | ncdtut1 | 控制设计示例 |
| | ncdtut2 | 系统辨识示例 |
| | paramdlg | 最优化参数帮助对话框 |
| | readncd | 与 read.m 内容相同 |
| | stepdlg | 阶跃响应帮助对话框 |

表 A-27 鲁棒控制工具箱（Robust Control Toolbox）

| 类 别 | 函数名称 | 功能描述 |
|---|---|---|
| 结构及分析函数 | aresolv | 求解广义连续 Riccati 方程 |
| | blkrsch | 由 cschur 得到块有序实 schur 形式 |
| | branch | 从树中提取一个分支 |
| | cschur | 通过复旋转得到有序复 schur 形式 |
| | daresolv | 求解广义离散 Riccati 方程 |
| | driccond | 计算离散 Riccati 方程的条件数 |
| | graft | 向树中增加一分支 |
| | iofc | 状态空间内外因子分解（列类型） |
| | iofr | 状态空间内外因子分解（行类型） |
| | istree | 辨识一个树型变量 |
| | mksys | 建立系统树变量 |
| | riccond | 计算连续 Riccati 方程的条件数 |
| | sfl | 左边频谱分解 |

（续表）

| 类　别 | 函 数 名 称 | 功 能 描 述 |
|---|---|---|
| | sfr | 右边频谱分解 |
| | tree | 建立树变量 |
| | vrsys | 返回标准系统变量名 |
| | | |
| 模型函数 | augss | 系统状态空间模型增广 |
| | augtf | 系统传递函数模型增广 |
| | balmr | 截断均衡模型 |
| | bilin | 多变量双线性变换 |
| | bstschml | 简化均衡 BST 模型 |
| | bstschmr | 简化均衡 BST 模型 |
| | des2ss | 利用奇异值分解变系统为状态空间模型 |
| | imp2ss | 由脉冲响应实现状态空间模型 |
| | interc | 建立一般多变量内联系统 |
| | lftf | 线性分式变换 |
| | obalreal | 实现有序均衡 |
| | ohklmr | Hankel 极小化最优逼近 |
| | rschur | 简化 schur 模型 |
| | sectf | 扇形变换 |
| | stabproj | 稳定和逆稳定映射 |
| | slowfast | 慢/快分解 |
| | tfm2ss | 变传递函数模型为状态空间模型 |
| | | |
| 鲁棒控制方法函数 | dh2lqg | 离散 $H_2$ 鲁棒控制 |
| | dhinf | 离散 $H_\infty$ 鲁棒控制 |
| | h2lqg | 连续 $H_\infty$ 鲁棒控制 |
| | hinf | 连续 $H_2$ 鲁棒控制 |
| | hinfopt | 通过 $\gamma$ 迭[illegible]合成 |
| | lqg | LQG 最[illegible]成 |
| | ltru | 连续 LQG/LTR 控制合成（输入） |
| | ltry | 连续 LQG/LTR 控制合成（输出） |
| | normh2 | 计算 $H_2$ 范数 |
| | normhinf | 计算 $H_\infty$ 范数 |
| | youla | Youla 参数化 |
| | | |
| | accdemo | 设计弹簧质量标准演示示例 |
| | dintdemo | 双积分器系统的 $H_\infty$ 设计示例 |





（续表）

| 类　别 | 函数名称 | 功能描述 |
| --- | --- | --- |
| 小例函数 | hinfdemo | 大型空间结构的 $H_\infty$ 设计示例 |
| | ltrdemo | 飞机 LQR/LTR 设计示例 |
| | mudemo | μ控制合成示例 |
| | mudemo1 | μ控制合成示例 |
| | mrdemo | 鲁棒模型简化示例 |
| | rctdemo | 鲁棒控制工具箱演示 |

表 A-28　信号处理工具箱（Signal Processing Toolbox）

| 类　别 | 函数名称 | 功能描述 |
| --- | --- | --- |
| 系统变换函数 | bilinear | 双线性变换 |
| | convmtx | 卷积矩阵 |
| | czt | 线性调频 z 变换 |
| | dct | 离散余弦变换 |
| | dftmtx | 离散傅里叶变换矩阵 |
| | fft | 一维快速傅里叶变换 |
| | fft2 | 二维快速傅里叶变换 |
| | fftshift | 重排 FFT 的输出 |
| | hilbert | 希尔波特变换 |
| | idct | 逆离散余弦变换 |
| | ifft | 一维快速傅里叶逆变换 |
| | ifft2 | 二维快速傅里叶逆变换 |
| | impinvar | 冲击响应不变法实现模拟到数字滤波器的变换 |
| | poly2rc | 由多项式系数计算反射系数 |
| | rc2poly | 由反射系数计算多项式系数 |
| | residuez | Z 变换部分分式展开或留数计算 |
| | sos2ss | 变系统二阶分割形式为状态空间形式 |
| | sos2sf | 变系统二阶分割形式为传递函数形式 |
| | sos2zp | 变系统二阶分割形式为零极点形式 |
| | ss2sos | 变系统状态空间形式为二阶分割形式 |
| | ss2tf | 变系统状态空间形式为传递函数形式 |
| | ss2zp | 变系统状态空间形式为零极点形式 |
| | tf2ss | 变系统传递函数形式为状态空间形式 |
| | tf2zp | 变系统传递函数形式为零极点形式 |
| | zp2sos | 变系统零极点形式为二阶分割形式 |
| | zp2ss | 变系统零极点形式为状态空间形式 |

（续表）

| 类　别 | 函数名称 | 功能描述 |
|---|---|---|
| | zp2tf | 变系统零极点形式为传递函数形式 |
| | | |
| 滤波器分析与实现 | fftfilt | 用重叠相加法实现 FFT 滤波器 |
| | filter | 一维滤波器实现 |
| | filter2 | 二维滤波器实现 |
| | filtfilt | 实现零相位数字滤波器 |
| | filtic | 设置'filter'函数的初始条件 |
| | freqs | 模拟滤波器的频率响应 |
| | freqspace | 频响中的频率间隔 |
| | freqz | 数字滤波器频率响应 |
| | grpdelay | 平均滤波延迟 |
| | impz | 数字滤波器的冲击响应 |
| | zplane | 离散系统零极点图 |
| | | |
| 滤波器的设计 | besself | Bessel 模拟滤波器设计 |
| | besselap | Bessel 模拟低通滤波器 |
| | buttap | Butterworth 模拟低通滤波器 |
| | butter | Butterworth 滤波器设计 |
| | buttord | 确定 Butterworth 滤波器的阶数 |
| | cheblord | 确定 ChebyshevI型滤波器的阶数 |
| | chep1ap | ChebyshevI型模拟低通滤波器 |
| | chep2ap | ChebyshevII型模拟低通滤波器 |
| | cheb2ord | 确定 ChebyshevII型滤波器的阶数 |
| | cheby1 | ChebyshevI型滤波器设计 |
| | cheby2 | ChebyshevII型滤波器设计 |
| | ellip | 设计椭圆滤波器 |
| | ellipap | 椭圆模拟低通滤波器 |
| | ellipord | 确定椭圆滤波器的阶数 |
| | fir1 | 使用窗函数设计滤波器（标准类型） |
| | fir2 | 使用窗函数设计滤波器（任意类型） |
| | firls | 最小二乘设计线性相位滤波器 |
| | fsamp2 | 由频率采样设计二维 FIR 滤波器 |
| | fspecial | 特殊二维滤波器设计 |
| | ftrans2 | 由频率变换设计二维 FIR 维滤波器 |
| | fwind1 | 使用一维窗函数设计 FIR 滤波器 |
| | fwind2 | 使用二维窗函数设计 FIR 滤波器 |





（续表）

| 类　别 | 函 数 名 称 | 功 能 描 述 |
|---|---|---|
| | intfilt | 内插 FIR 滤波器设计 |
| | remex | Parks-McCellan 最优 FIR 滤波器设计 |
| | remezord | 估计 Parks-McCellan 最优 FIR 滤波器阶数 |
| | yulewalk | 递归滤波器设计 |
| | | |
| 信号处理函数 | cceps | 倒谱分析和最小相位重构 |
| | cohere | 相关函数幅值平方估计 |
| | colfilt | 局部非线性滤波 |
| | conv2 | 二维卷积 |
| | cov | 协方差矩阵 |
| | corrcoef | 相关系数矩阵 |
| | csd | 互谱密度估计 |
| | deconv | 反卷积 |
| | decimate | 低通滤波后低速率重新采样 |
| | demod | 通信仿真中的解调 |
| | detrend | 消除线性趋势 |
| | interp | 使用低通内插高速率重新采样 |
| | medfilt1 | 一维中值滤波 |
| | medfilt2 | 二维中值滤波 |
| | mfilter2 | 屏蔽滤波 |
| | modulate | 通信仿真中的调制 |
| | nlfilter | 局部平滑 |
| | psd | 信号功率谱密度估计 |
| | rceps | 实倒谱和最小相位重构 |
| | specgram | 频谱计算 |
| | tfe | 由输入输出估计传递函数 |
| | wiener2 | 自适应二维维纳滤波 |
| | xcorr | 互相关函数估计 |
| | xcorr2 | 二维互相关函数估计 |
| | xcov | 互协方差函数估计 |
| | | |
| 窗函数 | boxcar | 矩形窗 |
| | bartlett | 巴特利特窗 |
| | blackman | 布莱克曼窗 |
| | chebwin | 切比雪夫窗 |
| | hamming | 哈明窗 |

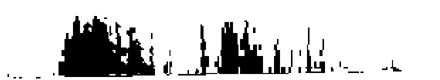
（续表）

| 类　别 | 函 数 名 称 | 功 能 描 述 |
|---|---|---|
| | hanning | 汉宁窗 |
| | kaiser | Kaiser 窗 |
| | triang | 三角窗 |
| | | |
| 其他函数 | invfreqs | 滤波器频率响应的模拟最小二乘拟合 |
| | invfreqz | 滤波器频率响应的离散最小二乘拟合 |
| | prony | 使用 prony 方法设计时域 IIR 滤波器 |
| | levinson | Levinson-Durbin 递归 |
| | lpc | 线性预测系数 |
| | lp2bp | 低通到带通模拟滤波器变换 |
| | lp2bs | 低通到带阻模拟滤波器变换 |
| | lp2hp | 低通到高通模拟滤波器变换 |
| | lp2lp | 低通到低通模拟滤波器变换 |

**表 A-29　图像处理工具箱（Image Processing Toolbox）**

| 类　别 | 函 数 名 称 | 功 能 描 述 |
|---|---|---|
| 图像颜色处理函数 | brighten | 加亮颜色板 |
| | bwarea | 二进制图像中的目标区域 |
| | cmunique | 寻找惟一的颜色及相应的图像 |
| | cmgamdef | 默认的γ校正表 |
| | cmpermute | 置换颜色板位置 |
| | cmgamma | γ校正颜色表 |
| | dilate | 加重二进制图像 |
| | dither | 是用颤抖算法变换图像 |
| | edge | 提取图像边界 |
| | erode | 冲淡二进制图像 |
| | grayslice | 使用门限由强度图像生成索引图像 |
| | histeq | 均衡直方图 |
| | hsv2rgb | 变 HSV 值为 RGB 值 |
| | imadjust | 调整并增强图像强度 |
| | imapprox | 用更少的颜色进行图像逼近 |
| | imhist | 显示图像数据的直方图 |
| | impixel | 给定像素点的颜色值 |
| | improfile | 图像轮廓强度 |
| | | |





（续表）

| 类　别 | 函数名称 | 功能描述 |
|---|---|---|
| 图像转换函数 | gray2ind | 变灰度图像为浮标图像 |
| | hsv2rgb | 变 HSV 值为 RGB 值 |
| | im2bw | 变图像为黑白图形 |
| | imcrop | 修剪图像 |
| | imresize | 改变图像大小 |
| | imrotate | 旋转图像 |
| | imslice | 获取图像块的索引 |
| | imzoom | 放大或缩小图像 |
| | ind2gray | 变浮标图像为 RGB 图像 |
| | mat2gray | 变矩阵为强度图像 |
| | ntsc2rgb | 变 NTSC 值为 RGB 值 |
| | rgb2gray | 变 RGB 值为灰度值 |
| | rgb2hsv | 变 RGB 值为 HSV 值 |
| | rgb2ntsc | 变 RGB 值为 NTSC 值 |
| | rgbplot | 绘制 RGB 颜色图形 |
| | | |
| 控制图像显示函数 | colorbar | 显示颜色条 |
| | colormap | 颜色查找表 |
| | gray | 线性灰度颜色板 |
| | hsv | HSV 颜色板 |
| | hot | “黑红黄白”颜色板 |
| | jet | HSV 变量 |
| | image | 显示浮标图像 |
| | imagesc | 扫描图像数据并显示 |
| | imcontour | 图像等高线 |
| | immovie | 制作动画图像 |
| | montage | 按矩阵剪辑方式显示图像 |
| | subimage | 在单一窗口显示多个图像 |
| | truesize | 调整图像的显示大小 |
| | warp | 以纹理表面方式显示图像 |
| | | |
| | bmpread | 由磁盘读取.BMP 文件 |
| | bmpwrite | 向磁盘写入.BMP 文件 |
| | getimage | 从坐标系中读取图像数据 |
| | gifread | 由磁盘读取.GIF 文件 |
| | gifwrite | 向磁盘写入.GIF 文件 |

（续表）

| 类　别 | 函数名称 | 功能描述 |
|---|---|---|
| 图像输入输出操作函数 | hdfpeek | 列出.HDF 文件中的目标标记/参考对 |
| | hdfread | 读取.HDF 文件中的数据 |
| | hdfwrite | 向.HDF 文件中写入数据 |
| | isbw | 图像为黑白图像时，返回真 |
| | isgray | 图像为灰度图像时，返回真 |
| | isind | 图像为浮标图像时，返回真 |
| | pcxread | 由磁盘读取.PCX 文件 |
| | pcxwrite | 向磁盘写入.PCX 文件 |
| | tiffread | 由磁盘读取.TIFF 文件 |
| | tiffwrite | 向磁盘写入.TIFF 文件 |
| | xwdread | 由磁盘读取.XWD 文件 |
| | xwdwrite | 向磁盘写入.XWD 文件 |
| | | |
| 示例函数 | bwmorph.mat | 对二进制图像进行形态操作的示例 |
| | dctdemo | 二维离散余弦变换图像压缩演示 |
| | firdemo | 二维 FIR 滤波器演示 |
| | forest.mat | Carmanah Old Growth Forest 的扫描相片 |
| | imdemo | 一般图像处理演示 |
| | mri.mat | 人体心脏的磁性共振图像 |
| | nlfdemo | 二维非线性滤波演示 |
| | trees.mat | 树的扫描图像 |
| | | |
| 其他函数 | cumsum3d | 三维矩阵封装成二维矩阵时的累计和 |
| | dctmtx2 | 二维离散余弦变换矩阵 |
| | ditherc | 图像颤抖算法的.MEX 文件 |
| | elem3d | 三维矩阵封装成二维矩阵的元素位置 |
| | getline | 用鼠标选择线 |
| | getpts | 用鼠标选择点 |
| | getrect | 用鼠标选择矩形 |
| | gif | 压缩 GIF 数据 |
| | hdfreadc | 读取.HDF 文件的.MEX 文件 |
| | hdfpeekc | 搜索.HDF 文件的.MEX 文件 |
| | hdfwc | 写.HDF 文件的.MEX 文件 |
| | im2gray | 变图像为灰度 |
| | imhistc | 图像直方图计算的.MEX 文件 |
| | ndx3d | 三维矩阵封装成二维矩阵的索引 |

（续表）

| 类　别 | 函数名称 | 功能描述 |
|---|---|---|
| | rgb2im | 变 RGB 图像为强度图像 |
| | rle | 压缩编码数据 |
| | tiff | 压缩 TIFF 编码数据 |
| | vmquant | 与彩色量化.MEX 文件接口的.M 文件 |
| | waitbar | 显示等待条 |

表 A-30　最优化工具箱（Optimization Toolbox）

| 类　别 | 函数名称 | 功能描述 |
|---|---|---|
| 极小化函数 | constr | |
| | fgoalattain | 求解多目标规划的优化问题 |
| | fmin | 求解单变量函数的极小值 |
| | fminbnd | 求解边界约束条件下的非线性极小值 |
| | fmincon | 求解约束条件下的非线性极小值 |
| | fminimax | 求解最小最大极值 |
| | fmins | 同 fminsearch |
| | fminsearch | 求解无约束条件下的非线性极小值 |
| | fminu | fminunc |
| | fminunc | 求解多变量函数的极小值 |
| | fseminf | 求解半无穷条件下的极小值 |
| | linprog | 求解线性规划问题 |
| | quadprog | 求解二次规划问题 |
| | | |
| 方程求解函数 | fsolve | 求解非线性方程 |
| | fzero | 求解标量非线性方程 |
| | | |
| 最小二乘优化函数 | lsqlin | 求解约束条件下的线性最小平方问题 |
| | lsqcurvefit | 求解非线性曲线拟合问题 |
| | lsqnonlin | 求解非线性最小平方问题 |
| | lsqnonneg | 求解非负线性最小平方问题 |
| | | |
| 插值函数 | cubic | 4 点内插值估计极大值 |
| | cubici1 | 2 点内插值和梯度估计极小值 |
| | cubici2 | 3 点内插值和 1 梯度 |
| | cubici3 | 2 点内插值和梯度估计步长和极小值 |
| | quad2 | 3 点内插值估计极大值 |

（续表）

| 类　别 | 函数名称 | 功能描述 |
|---|---|---|
| | quadinter | 3 点内插值估计极小值 |
| | | |
| 内部的实用函数程序 | eigfun | 返回分类特征的函数（用于 goaldemo） |
| | elimone | 消除一个变量（用于 dfildemo） |
| | filtfun | 返回频率响应和根（用于 dfildemo） |
| | findmax | 在数据向量中内插极大值 |
| | findmax2 | 在数据矩阵中内插极大值 |
| | filtfun2 | 返回频率响应和根 |
| | fitfun | 返回数据与计算值之间的误差（用于 fitdemo） |
| | fitfun2 | 返回数据拟合的误差（用于 datdemo） |
| | goalfun | 目标逼近问题的转换函数 |
| | goalgra | 目标逼近问题的梯度更换函数 |
| | graderr | 检查优化中的梯度不一致性 |
| | lsint | 最小 乘优化程序的初始化函数 |
| | optint | 无约束最优化程序的初始化函数 |
| | searchq | 线性搜索程序 |
| | semifun | 变半无穷优化问题的转化程序 |
| | toptim | 最优化测试组 |
| | toptimf | 最优化测试组的测试函数 |
| | toptimg | 最优化测试组的测试函数的测试梯度 |
| | v2sort | 分成两组向量并删除丢失的元素 |
| | | |
| 演示示例 | bandemo | 香蕉函数的极小化示例 |
| | datdemo | 数据拟和示例 |
| | dfildemo | 有限精度滤波器设计示例 |
| | goaldemo | 目标逼近优化示例 |
| | optdemo | 菜单演示示例 |
| | tutdemo | 启动教程 |

**表 A-31　神经网络工具箱（Neural Toolbox）**

| 类　别 | 函数名称 | 功能描述 |
|---|---|---|
| 神经网络设计函数 | solvehop | 设计 Hopfeild 神经网络 |
| | solvelin | 设计线性神经网络 |
| | solverb | 设计径向基神经网络 |
| | solvebe | 设计精确的径向基神经网络 |

（续表）

| 类 别 | 函数名称 | 功能描述 |
|---|---|---|
| 学习规则函数 | learnbp | 误差反向传播学习规则 |
| | learnbpm | 带动量项的误差反向传播学习规则 |
| | learnh | Hebb 学习规则 |
| | learnhd | 权值退化的 Hebb 学习规则 |
| | learnis | 内星学习规则 |
| | learnk | Kohonen 学习规则 |
| | learnlm | Levenberg-Marquardt 学习规则 |
| | learnlvq | 学习向量量化学习规则 |
| | learnos | 外星学习规则 |
| | learnp | 感知器学习规则 |
| | learnpn | 归一化的感知器学习规则 |
| | learnwh | Widrow-Hoff 学习规则 |
| | | |
| 神经网络训练函数 | trainbp | 使用误差反向传播训练网络 |
| | trainbpx | 使用快速误差反向传播训练网络 |
| | trainc | 训练竞争层网络 |
| | trainelm | 训练 Elman 递归网络 |
| | trainlvq | 训练 LVQ 网络 |
| | trainp | 使用感知规则训练感知层 |
| | trainpn | 使用归一化感知规则训练感知层 |
| | trainsm | 使用 Kohonen 规则训练自组织映射 |
| | trainwh | 使用 Widrow-Hoff 规则训练线性层 |
| | | |
| 绘图函数 | barerr | 绘制输出向量的误差条形图 |
| | errsurf | 计算误差曲面 |
| | hintonw | 绘制权值图 |
| | hintonwb | 绘制权值和偏差图 |
| | plotep | 在误差曲面上绘制权和基位置图 |
| | plotfa | 绘制数据点及网络函数逼近 |
| | ploterr | 绘制网络平方和误差与时间的关系图 |
| | plotes | 绘制误差曲面图 |
| | plotpv | 绘出感知器输入/目标向量 |
| | plotsm | 绘制自组织映射图 |
| | plottr | 绘制误差、学习速率与时间的关系图 |
| | plotvec | 用不同颜色绘制向量 |
| | | |

（续表）

| 类　别 | 函数名称 | 功能描述 |
|---|---|---|
| 神经网络仿真函数 | simuc | 竞争层仿真 |
| | simuelm | Elman 递归神经网络仿真 |
| | simuff | 前向神经网络仿真 |
| | simuhop | Hopfield 神经网络仿真 |
| | simulin | 线性层仿真 |
| | simup | 感知器仿真 |
| | simurb | 径向基神经网络仿真 |
| | simusm | 自组织映射仿真 |
| | | |
| 初始化函数 | deltalin | PURELIN 神经元的δ函数 |
| | deltalog | LOGSIG 神经元的δ函数 |
| | deltatan | TRANSIG 神经元的δ函数 |
| | initc | 竞争层初始化 |
| | initelm | Elman 递归神经网络初始化 |
| | initff | 最多三层的前向神经网络初始化 |
| | initlin | 线性层初始化 |
| | initlvq | LVQ 神经网络初始化 |
| | initp | 感知层初始化 |
| | initsm | 自组织映射初始化 |
| | midpoint | 中间权值初始化 |
| | nwlog | 产生 LOGSIG 神经元的 Nguyen-Widrow 随机数 |
| | nwtan | 产生 TRANSIG 神经元的 Nguyen-Widrow 随机数 |
| | randnc | 生成归一化列随机数 |
| | randnr | 生成归一化行随机数 |
| | rands | 生成对称随机数 |
| | | |
| 传递函数 | compet | 竞争层传递函数 |
| | hardlim | 限幅传递函数 |
| | hardlims | 对称限幅传递函数 |
| | logsig | 对数 S 型传递函数 |
| | purelin | 线性传递函数 |
| | radbas | 径向基传递函数 |
| | satlins | 对称饱和线性传递函数 |
| | tansig | 正切 S 型传递函数 |





表 A-32　模糊逻辑工具箱（Fuzzy Logic Toolbox）

| 类　别 | 函数名称 | 功能描述 |
|---|---|---|
| 隶属度函数 | dsigmf | 两个 S 函数的差构成的隶属度函数 |
| | gauss2mf | 双边高斯曲线隶属度函数 |
| | gbellmf | 广义钟形隶属度函数 |
| | guassmf | 高斯曲线隶属度函数 |
| | pimf | π形隶属度函数 |
| | psigmf | 两个 S 函数的积构成的隶属度函数 |
| | smf | S 形曲线隶属度函数 |
| | sigmf | Sigmoid 形曲线隶属度函数 |
| | trapmf | 梯形隶属度函数 |
| | trimf | 三角形隶属度函数 |
| | zmf | Z 形隶属度函数 |
| | | |
| 模糊推理系统函数 | addmf | 向 FIS（模糊推理系统）中添加隶属度函数 |
| | addrule | 向 FIS 中添加规则 |
| | addvar | 向 FIS 中添加变量 |
| | defuzz | 去模糊隶属度函数 |
| | evalfis | 完成模糊推理计算 |
| | evalmf | 隶属度函数计算 |
| | fuzzy | 模糊逻辑工具箱 |
| | gensurf | 生成 FIS 输出曲面 |
| | getfis | 获取模糊系统的特性 |
| | mf2mf | 隶属度函数之间的参数转换 |
| | newfis | 生成新的 FIS |
| | parsrule | 模糊分析规则 |
| | plotfis | 显示 FIS 输入/输出图 |
| | readfis | 由磁盘装入 FIS |
| | rmmf | 从 FIS 删除隶属度函数 |
| | rmvar | 从 FIS 删除变量 |
| | setfis | 设置模糊系统特性 |
| | showfis | 显示带注释的 FIS |
| | showrule | 显示 FIS 规则 |
| | writefis | 在磁盘中保存 FIS |
| | | |
| | anfisedit | ANFIS 训练和测试用户接口工具 |

（续表）

| 类　别 | 函数名称 | 功能描述 |
|---|---|---|
| GUI 编辑器 | findcluster | 族用户接口工具 |
| | fuzzy | 基本 FIS 编辑器 |
| | mfedit | 隶属度函数编辑器 |
| | ruleedit | 规则编辑器和分析程序 |
| | ruleview | 规则观察器和模糊推理框图 |
| | surfview | 输出曲面观察器 |
| | | |
| 先进技术 | anfis | Sugeno-type FIS 的训练程序 |
| | fcm | 利用模糊 C 平均聚集方法找出族 |
| | genfis1 | 使用一般方法生成 FIS 矩阵 |
| | genfis2 | 使用减法聚集方法生成 FIS 矩阵 |
| | subclust | 使用减法聚集方法估计族中心 |
| | | |
| 其他函数 | convertfis | 将 v1.0 模糊矩阵转化为 v2.0 模糊结构 |
| | discfis | 离散化 FIS |
| | evalmmf | 估计多个隶属度函数 |
| | findrow | 找出匹配输入的矩阵的行 |
| | fstrvcat | 可变大小的连接矩阵 |
| | fuzarich | 模糊数学函数 |
| | genparam | 为 ANFIS 学习生成初精度的参数 |
| | sugmax | Sugemo 系统的最大输出范围 |

表 A-33　系统辨识工具箱（System Identification Toolbox）

| 类　别 | 函数名称 | 功能描述 |
|---|---|---|
| 参数估计函数 | ar | 使用各种方法计算信号的 AR 模型 |
| | armax | 估计 ARMAX 模型预测误差 |
| | arx | ARX 模型的最小二乘估计 |
| | bj | 估计 Ox-Jenkins 模型的预测误差 |
| | canstart | 具有初始参数估计的状态空间模形 |
| | covf | 估计数据矩阵的协方差矩阵 |
| | cra | 应用相关分析估计冲击响应 |
| | etfe | 估计经验传递函数和周期图 |
| | idsimsd | 说明仿真模型响应中的不确定性 |
| | ivar | 计算时间序列的 AR 部分的IV估计 |
| | ivx | 估计 ARX 模型的作用变量 |

（续表）

| 类　别 | 函数名称 | 功能描述 |
| --- | --- | --- |
| 参数估计函数 | iv4 | ARX 模型的近似最优IV估计 |
| | oe | 估计输出误差模型的预测误差 |
| | pem | 一般线性模型的预测误差估计 |
| | rarx | ARX 模型的递归估计 |
| | rarmax | ARMAX 模型的递归估计 |
| | rbj | Box-Jenkins 模型的递归估计 |
| | roe | 输出误差模型的递归估计 |
| | rpem | 一般模型的递归估计 |
| | rplr | 一般模型的 PLR 递归估计 |
| | segment | 分段数据并跟踪快变系统 |
| | th2ff | 计算模型频率函数和标准偏差 |
| | th2zp | 计算模型的零极点、静态增益和标准偏差 |
| | | |
| 模型结构函数 | arx2th | 构建 ARX 模型的 Θ 格式 |
| | arxstruc | 计算 ARX 模型类的损失函数 |
| | canform | 正则形模型函数 |
| | fixpar | 修正状态空间和 ARX 模型中的参数 |
| | ivstruc | 计算单输出 ARX 模型类的输出误差拟合 |
| | mf2th | 将用户定义的模型结构封装入 Θ 模型格式 |
| | modstruc | 在 ms2th 函数中使用的模型结构 |
| | ms2th | 将标准状态空间参数封装入 Θ 格式中 |
| | poly2th | 由给定多项式产生 Θ 矩阵 |
| | selstruc | 根据各种准则选择模型结构 |
| | sett | 在 Θ 结构中设置取样间隔 |
| | struc | 生成 ARXSTRUC 和 INVSTRUC 的典型结构矩阵 |
| | thinit | 为迭代估计初始化参数 |
| | unfixpar | 放松状态空间和 ARX 模型的参数 |
| | | |
| 模型处理函数 | bodeplot | 绘制传递函数的波特图和频谱图 |
| | compare | 比较仿真预测的输出与测量输出 |
| | dtrend | 消除数据集中的趋势项 |
| | ffplot | 绘制传递函数的波特图和频谱图 |
| | getff | 选择频率函数 |
| | getncap | 获取数据点数和参数个数 |
| | gett | 获取模型的采样间隔 |

（续表）

| 类　别 | 函数名称 | 功能描述 |
|---|---|---|
| 模型处理函数 | getzp | 获取由 TH2ZP 函数产生的零极点格式中的零极点 |
| | idfilt | 用巴特沃斯滤波器对数据进行滤波 |
| | idplot | 绘制输入输出数据图形 |
| | idsim | 仿真给定的动态系统 |
| | nyqplot | 绘制频率函数的 Nyquist 图 |
| | pe | 计算预测误差 |
| | predict | *M* 步超前预测 |
| | present | 显示一个参数模型 |
| | resid | 计算和测试某模型的留数 |
| | zpplot | 绘制零极点图 |
| | | |
| 模型转换函数 | th2arx | 变 Θ 格式模型为 ARX 模型 |
| | th2ff | 计算模型的频率函数和标准偏差 |
| | th2par | 变 Θ 格式为参数和协方差矩阵 |
| | th2poly | 求给定模型相应的多项式 |
| | th2ss | 变 Θ 格式为状态空间表示 |
| | th2tf | 变 Θ 格式为传递函数表示 |
| | th2zp | 变 Θ 格式为零极点表示 |
| | thc2thd | 变连续模型为离散模型 |
| | thd2thc | 变离散模型为连续模型 |

**表 A-34　小波分析工具箱（Wavelet Toolbox）**

| 类　别 | 函数名称 | 功能描述 |
|---|---|---|
| 一般函数 | biorfilt | 二次正交小波滤波器集 |
| | dyaddown | 二分下采样函数 |
| | dyadup | 二分上采样函数 |
| | intwave | 集成小波函数 |
| | orthfilt | 对称小波滤波器集 |
| | qmf | 积分镜像滤波器 |
| | wavefun | 小波尺度函数 |
| | wavemngr | 小波管理器 |
| | wfilters | 小波滤波器 |
| | wmaxlev | 小波分解的最大级数 |
| | | |

（续表）

| 类别 | 函数名称 | 功能描述 |
|---|---|---|
| 小波族函数 | biorwavf | 二次样条小波滤波器 |
| | cgauwavf | 复高斯小波 |
| | cmorwavf | 复 Morlet 小波 |
| | coifwavf | Coiflet 小波滤波器 |
| | dbaux | 计算 Daubechies 小波滤波器 |
| | dbwavf | Daubechies 小波滤波器 |
| | fbspwavf | 复频率二次样条小波 |
| | gauswavf | Gauss 小波 |
| | mexihat | 墨西哥草帽小波 |
| | meyer | Meyer 小波 |
| | meyeraux | Meyer 小波辅助函数 |
| | morlet | Morlet 小波 |
| | rbiowavf | 反二次样条小波滤波器 |
| | shanwavf | 复 Shannon 小波 |
| | symaux | Symlet 小波滤波器计算 |
| | symwavf | Symlet 小波滤波器 |
| | | |
| 一维小波函数 | cwt | 实或复连续小波系数 |
| | appcoef | 离散一维逼近系数 |
| | detcoef | 离散一维细节系数 |
| | dwt | 离散一维小波一级变换 |
| | dwtmode | 离散小波变换扩展模式 |
| | idwt | 离散一维小波一级逆变换 |
| | upcoef | 由一维小波系数直接重构信号 |
| | upwlev | 一维小波分解的一级信号重构 |
| | wavedec | 一维小波多级分解 |
| | waverec | 一维小波多级重构 |
| | wrcoef | 由一维小波系数重构单级分支 |
| | | |
| 二维小波函数 | appcoef2 | 离散二维小波逼近系数 |
| | detcoef2 | 离散二维小波细节函数 |
| | dwt2 | 一级离散二维小波变换 |
| | dwtmode | 离散小波转换扩展模式 |
| | idwt2 | 一级离散二维小波逆变换 |
| | upcoef2 | 由二维小波系数直接重构信号 |
| | upwlev2 | 二维小波分解的一级信号重构 |

（续表）

| 类　别 | 函数名称 | 功能描述 |
|---|---|---|
| | wavedec2 | 二维小波多级分解 |
| | waverec2 | 二维小波多级重构 |
| | wrcoef2 | 由二维小波系数重构单级分支 |
| | | |
| 小波包函数 | bestlevt | 最优级树（小波包） |
| | besttree | 最优树（小波包） |
| | entrupd | 熵更新（小波包） |
| | wentropy | 熵（小波包） |
| | wp2wtree | 从小波包树抽取小波树 |
| | wpcoef | 小波包系数 |
| | wpcutree | 裁减小波包树 |
| | wpdec | 一维小波包分解 |
| | wpdec2 | 二维小波包分解 |
| | wpfun | 小波包函数 |
| | wpjoin | 重新分解小波包 |
| | wprcoef | 重构小波包系数 |
| | wprec | 一维小波包重构 |
| | wprec2 | 二维小波包重构 |
| | wpsplt | 小波包分解 |
| | | |
| | | |
| 离散静态小波变换 | iswt | 一维离散静态小波反变换 |
| | iswt2 | 二维离散静态小波反变换 |
| | swt | 一维离散静态小波变换 |
| | swt2 | 二维离散静态小波变换 |
| | | |
| 噪声分离和信号图像压缩 | ddencmp | 噪声分离或压缩的默认值 |
| | thselect | 选择噪声分离的门限值 |
| | wbmpen | 一维或二维小波噪声分离的惩罚门限 |
| | wdcbm | 使用 Birge-Massart 的一维小波的门限 |
| | wbcbm2 | 使用 Birge-Massart 的二维小波的门限 |
| | wden | 利用小波的自动一维噪声分解 |
| | wdencmp | 利用小波的噪声分离或压缩 |
| | wnoise | 生成含有噪声的小波测试数据 |
| | wnoisest | 估计一维小波系数的噪声 |
| | wpbmpen | 小波包噪声分离的惩罚门限 |

（续表）

| 类　别 | 函数名称 | 功能描述 |
| --- | --- | --- |
| | wpdencmp | 利用小波包的噪声分离或压缩 |
| | wpthcoef | 小波包系数的门限 |
| | wthcoef | 一维小波系数的门限 |
| | wthcoef2 | 二维小波系数的门限 |
| | wthresh | 设置软和硬门限 |
| | wthrmngr | 门限设置管理器 |

# 附录 B　MATLAB 6.5 的新特性

## B.1　Simulink 5.0 的新特性

Simulink 是一种框图模型建模环境，用于动态系统的仿真、性能评估，并进行控制系统、DSP 系统和通信系统的设计。该软件在 GUI 和运行引擎方面的改进有：

- 线形化的程序调试器，使错误分析及调试操作简单起来；
- 支持大多数 Simulink 模块间以矩阵形式交换数据；
- 支持更高速的基于帧信号处理的 DSP 应用；
- 支持大模型开发的新特性，包括更新的工具栏选项和增强的行编辑功能；
- 集成的查找对话框工具方便了元件在 Simulink 模型和（或）状态流图中的搜索；
- 增加了 Simulink 数据对象，支持用户自定义数据结构；
- 新的航空库及示例；
- 提高了库浏览器和模型浏览器的使用性能；
- 废除了库连接，以方便库模型块的编辑；
- 增强了可配置子系统，便于设计模型向实际实现转化；
- 具有筛选算法的高级查表模块，使仿真和代码生成更加快速、精确、灵活；
- 单一视窗模式节省了宝贵的屏幕空间。

## B.2　MathWorks Release 13 新产品

1．**航空工具箱**（Aerospace Blockset）

航空工具箱以仿真为基础，提供具体的工具，用于飞机、飞船、导弹及推进系统和子系统的建模、集成和仿真。

- 对于宇航式飞行器，提供了推进、控制系统、系统动力学及激励控制箱；
- 用航空工具箱的组件提供了六自由度和三自由度的实例模型；
- 包含重力、空气和风的环境模型；
- 使用空间表示转换模块增强了坐标转换功能；
- 能实现物理特性的单元转换；
- 使用 Handle Graphics 技术，提供了动画模块，实现三自由度和六自由度的动画演示。

2．**曲线拟合工具箱**（Curve Fitting Toolbox）

曲线拟合工具箱提供了数据预处理的各种方法，比如生成、比较、分析和管理模型。

所有的功能既可以以命令行来实现，也可以以图形用户界面来实现。

- 预处理方法，包括数据的标定、截断、滤波和奇异值的消除等；
- 扩展了线形和非线形参数拟合模型库，对于非线形模型，起始点和参数的计算程序都进行了优化；
- 有各种各样的参数和非参数模型，包括最小二乘法、加权最小二乘法和鲁棒拟合过程（包括有参数边界限制和没有参数边界限制）；
- 自定义参数和非参数模型扩展；
- 用样条和内插法实现非参数拟合；
- 拟合数据的插补、外推、微分和积分

3．Motorola MPC555 嵌入式目标（Embedded Target for Motorola MPC555）

Motorola MPC555 嵌入式目标可以将由 MATLAB 实时工作空间编码器生成的代码直接配置到 MPC555 微控制器。Motorola MPC555 嵌入式目标依靠 MATLAB 实时工作空间编码器来为 Motorola MPC555 生成和提供具体的代码。

- 为 MPC555 提供快速原型和产品代码生成；
- 针对 Motorola MPC555 MIOS，QADC 和 TouCAN 设置驱动块；
- 通过板载调试器（onboard debugger）或 CAN（controller area network）下载应用到 Motorola MPC555；
- 通过处理器以回路联合仿真的方式验证算法代码；
- 生成详细的 HTML 代码生成报告，包括 RAM/ROM 信息。

4．C6000 DSP 平台嵌入式目标（Embedded Target for C6000 DSP Platform）

简化 Texas Instruments DSP 软件设计和分析流程，直接从 MathWorks 环境生成高效的代码。你可以使用 Simulink、DSP Blockset 和 Communications Blockset 中的块开发层次化的 DSP 算法框图模型，然后通过 Real-Time Workshop 生成无歧义的可执行算法，并可以进一步由 DSP 软件工程师优化。

- 自动生成支持 Texas Instruments DSP 目标的代码；
- 无需 DSP 编程，就可以实现验证算法；
- 生成 Code Composer Studio 项目格式的带注解的 C 代码。

5．MATLAB COM Builder

MATLAB COM Builder 可以很容易地将 MATLAB 算法转化为 COM 对象，可以用于任何基于 COM 的应用。

- 通过简便易用的 GUI 将 MATLAB 算法转换成 COM 对象；
- 生成 COM 对象，可以供 Visual Basic、C/C++、Microsoft Excel 和其他任何基于 COM 的应用使用；
- 允许修改和查看 MATLAB 生成的代码，可以对你所引用的函数有更好的理解；
- 通过 MATLAB 生成的 COM 对象可以自由发布。

6．MATLAB Excel Builder

MATLAB Excel Builder 使得很容易将复杂的 MATLAB 算法转化为独立的 Excel 插件

（add-ins），用户可以利用以矩阵为基础且灵活的 MATLAB 编程环境，以及拥有大量可用数学和图像函数的优点，从而迅速建立计算更深入的模型。

- 通过简便易用的 GUI 将 MATLAB 算法转换成 Excel 插件（add-in）；
- 自动生成 .dll 和 Visual Basic 应用文件，并且可以输入到 Excel 中；
- 生成的插件函数比 Visual Basic for Applications（VBA）生成的快 95%；
- 允许修改和查看 MATLAB 生成的代码，可以对你所引用的函数背后的逻辑有更好的理解。

7．Code Composer Studio 开发工具的 MATLAB 连接

简化 Texas Instruments DSP 软件设计和分析流程，实现在 IT 软件开发环境、实时 DSP 硬件和 MATLAB 之间的通信。算法开发人员、系统设计人员和嵌入工程师通过此工具可以在 MATLAB 中测试、验证和可视化 DSP 软件，消除了 DSP 算法研究和实现之间的缺口。

- 提供 M 文件功能，允许你对数据传输和验证任务进行定制和自动化；
- 在 MATLAB 和 Texas Instruments DSP 设备之间提供数据传输能力，而不必停止目标应用上执行的任务；
- 硬件对象和 Code Composer Studio 帮助测试、验证和可视化 DSP 代码。

8．基于模型校准工具箱（Model-Based Calibration Toolbox）

基于模型校准工具箱提供的设计工具来校准功率系统。它建立在 MATLAB 高性能的计算环境以及仿真能力的基础上，减少了功率器的检测时间，提高了工程效率，节省了校准时间，因而有潜力改进功率系统性能和可靠性。

- 新的经典设计：Plackett-Burman 是两水平筛选设计（tow-level screening design），规则单纯形（Regular Simplex）是一个效率高的一阶设计方法；
- 使用更方便，可以将详细用户信息存储起来，可以在 Model Browser 中跟踪项目文件的历史列表；
- 新的上下文菜单，可以复制、删除和重命名模型；
- 在现有的所有模型绘图放大功能之上，增加了新的数据绘图放大工具；
- 在 CAGE 用户界面上浏览大型数据集，速度可以提高 40% 以上；
- 支持 Simulink 库中的高级查表类型。

9．机械仿真（SimMechanics）

它允许工程师在 Simulink 环境中仿真机械系统。应用 SimMechanics 可以直接建立机械部件的模型，仿真它们的运动及分析结果，而不需要推导数学方程。

- 增强的非线性方程求解器，仿真速度更快；
- 优化的模型线性化；
- 新的邻接（adjoining）特性，方便对相邻的其他物体的坐标系位置和方向的定义；
- 对于并联机械模型，用户可以选择和观察切断铰链（cut joints）；
- 图形用户界面增强，模型自动化功能增强。

具体内容可到网站 http://www.mathworks.com 上查阅。

# 附录C　MATLAB 6.5 安装问题指南

## C.1　MATLAB 6.5 为什么安装后不能启动

经常有朋友遇到这个问题，其实在 Windows 98 或其第 2 版下安装 MATLAB 6.5 应该没有问题（也有一部分可能会在安装了 IE 5.5 或更高版本并更新了 Windows 的一些字库以后出现），而在 Windows Me 或 Windows 2000 下安装 MATLAB 6.5 不能启动的主要原因是它的部分中文字库和操作系统的字库重名造成的，下面给出几种可行的解决方法以供选择。

### 1. 更改区域设置

在控制面板里有“区域设置”一项，一般的中文 Windows 操作系统上默认的区域设置是“**中文（中国）**”，如图 C-1 所示。这里我们只要把它更改为“**英语（美国）**”之后重新启动计算机，MATLAB 6.5 就可以正常运行了。不用担心，改过区域设置之后，操作系统仍然是中文操作系统，并不会给我们的使用带来太多的麻烦（当然有些小的显示等方面的问题）。这也是最省事的一种解决方法了。

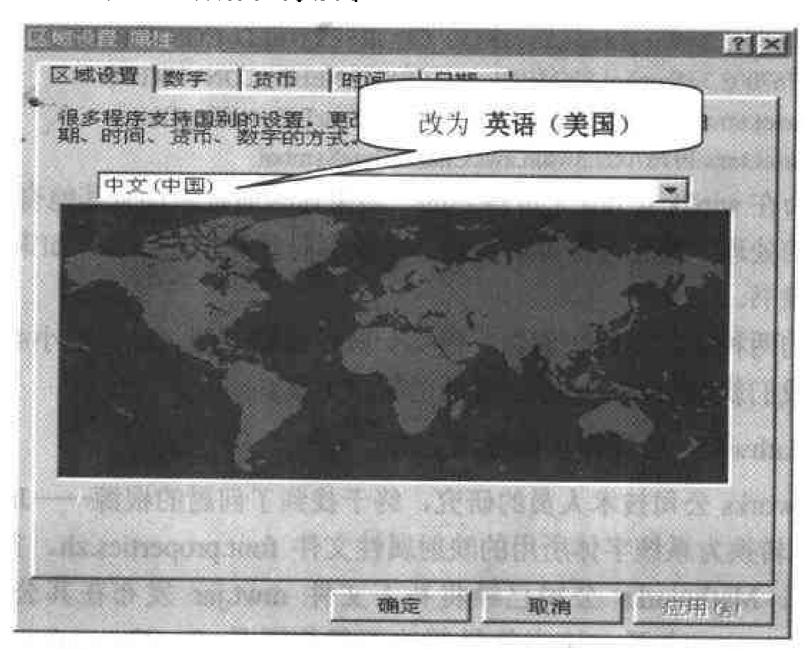



图 C-1　更改区域设置

### 2. 消除重名字库

可以直接在操作系统中将重名字库删除，或是在 MATLAB 6.5 的字体配置文件中改变重名字库的相关信息。前一种删除方法将导致该字库从操作系统中被完全删掉，其他应用

软件将也无法使用该字库，因此并不十分理想，不过去也非常方便。后一种方法，该字库没有从操作系统中删掉，只是在 MATLAB 6.5 的字体属性配置文件中改变了相关信息，因此并不影响其他应用软件使用这些字库。这两种删除方法共同的最核心的问题就在于要知道是哪一些字库重名了，而这往往需要去试验或是获得他人的经验。目前在一般的情况下，大约只有“新宋体”这个字库有重名的问题。第一种删除字体的方法也非常简单（在字体目录下将相应的文件删除即可），但一般不提倡使用。这里，我们就介绍如何更改 MATLAB 6.5 的字体配置文件。

首先，MATLAB 6.5 这个导致死机的字体属性配置文件是 font.properties.zh，这个文件在目录$MATLAB\sys\java\jre\win32\jre\lib\下（$MATLAB 表示 MATLAB 6.5 的安装目录）。我们可以用一个文本编辑器来编辑这个文件。

在 $MATLAB\sys\java\jre\win32\jre\lib\font.properties.zh 文件的最后加上：

```
# Font entry added by The MathWorks:"新宋体"
tmw36525210.0=\u65b0\u5b8b\u4f53,ANSI_CHARSET
tmw36525210.1=\u65b0\u5b8b\u4f53,GB2312_CHARSET
tmw36525210.2=WingDings,SYMBOL_CHARSET,NEED_CONVERTED
tmw36525210.3=Symbol,SYMBOL_CHARSET,NEED_CONVERTED
fontcharset.tmw36525210.2=sun.awt.windows.CharToByteWingDings
fontcharset.tmw36525210.3=sun.awt.CharToByteSymbol

# Font entry added by The MathWorks:"@新宋体"
tmw39767002.0=@\u65b0\u5b8b\u4f53,ANSI_CHARSET
tmw39767002.1=@\u65b0\u5b8b\u4f53,GB2312_CHARSET
tmw39767002.2=WingDings,SYMBOL_CHARSET,NEED_CONVERTED
tmw39767002.3=Symbol,SYMBOL_CHARSET,NEED_CONVERTED
fontcharset.tmw39767002.2=sun.awt.windows.CharToByteWingDings
fontcharset.tmw39767002.3=sun.awt.CharToByteSymbol
```

以上结果为在 win2k sc pro 上所得到的，如果你的系统中还有其他有问题的中文字库，需要进行同样的处理。经过检验可以找到一些没有问题的中文字库，如宋体、黑体、仿宋、楷体、幼圆、隶书。

这种方法的两种手段都比较麻烦，需要一些经验和试验，如果不小心还可能会出现一些问题，下面我们就介绍 Mathworks 公司的解决方案。

**3．使用 Mathworks 公司提供的技术支持**

经过 Mathworks 公司技术人员的研究，终于找到了问题的根源——Java 虚拟机用于将标准 Java 字体转换为系统字体所用的映射属性文件 font.properties.zh，并且映射程序中存在一个小 Bug。Mathworks 公司已经将补丁文件 mwt.jar 发布在其公司的网站 http://www.mathworks.com 上了，该文件最新的下载位置是 ftp://ftp.mathworks.com/pub/tech-support/solutions/s26990。下载了该文件后替换目录$MATLAB\java\jar\下的同名文件（当然最好对初始的文件做一个备份），然后，重新启动 MATLAB 6.5，问题就可以解决了。如果仍有一些其他问题，可以到其公司或其他一些相关网站上查询。



## C.2　安装时更新Java虚拟机的问题

这个问题一般比较偶然地出现在Windows 98和Windows NT环境中，有的时候因为安装其他软件时已经解决了这个问题，在安装MATLAB 6.5时就会比较顺利。但有时确实会出现要求更新Java虚拟机的提示消息框，这时我们应当更新我们的Java环境。其软件一般可以在MATLAB 6.5的安装盘中找到，如果要下载的软件，可以到网址http://www.microsoft.com/java/vm/dl_vm40.htm上下载相应的软件msvm.exe（对应于Windows 98/NT操作系统）。如果是Windows 2000操作系统，只需要安装升级补丁SP2。

## C.3　PDF文档的获取

国内的MATLAB 6.5软件大多数是下载的软件，因此帮助文件不全。下面的这两个网址可以给大家一点帮助。

国外：http://www.mathworks.com/access/helpdesk/help/helpdesk.shtml；

国内：http://science.fire.ustc.edu.cn/matlab6.html。

# 附录D 遗传算法中的部分函数代码

本附录给出了遗传算法实现中辅助调用的函数代码，读者可以结合“第5章”加以理解和应用。



本附录的文件代码来自于遗传算法工具箱GAOT和GATBX。

1．b2f.m

【功能】将二进制数转化为浮点数。

```
function [fval] = b2f(bval,bounds,bits)
scale=(bounds(:,2)-bounds(:,1))'./(2.^bits-1); %变量范围
numV=size(bounds,1);
cs=[0 cumsum(bits)];
for i=1:numV
   a=bval((cs(i)+1):cs(i+1));
   fval(i)=sum(2.^(size(a,2)-1:-1:0).*a)*scale(i)+bounds(i,1);
end
```

2．f2b.m

【功能】将浮点数转化为二进制数。

```
function [bval] = f2b(fval,bounds,bits)
scale=(2.^bits-1)./(bounds(:,2)-bounds(:,1))'; %变量范围
numV=size(bounds,1);
cs=[0 cumsum(bits)];
bval=[];
for i=1:numV
   fval(i)=(fval(i)-bounds(i,1)) * scale(i);
   bval=[bval rem(floor(fval(i)*pow2(1-bits(i):0)),2)];
end
```

3．nonUnifMutate.m

【功能】基于非均一概率分布进行非均一变异。

```
function [parent] = nonUnifMutate(parent,bounds,Ops)
cg=Ops(1);                        % 当前代
mg=Ops(3);                      %代的最大数目
b=Ops(4);                         % 形状参数
df = bounds(:,2) - bounds(:,1);   %变量范围
numVar = size(parent,2)-1;        %变量数目
% Pick a variable to mutate randomly from 1 to number of vars
mPoint = round(rand * (numVar-1)) + 1;
md = round(rand);                 % 选择变异方向
```

```
if md                                          %向上界变异
   newValue=parent(mPoint)+delta(cg,mg,bounds(mPoint,2)-parent(mPoint),b);
else                                       %向下界变异
   newValue=parent(mPoint)-delta(cg,mg,parent(mPoint)-bounds(mPoint,1),b);
end
parent(mPoint) = newValue;           % 得到一个子代个体
```

4. normGeomSelect.m

【功能】基于正态分布的序列选择函数

```
function[newPop] = normGeomSelect(oldPop,options)
q=options(2);                              % 选择最优的概率
e = size(oldPop,2);                        % xZome 的长度，等于 numVars+fit
n = size(oldPop,1);                    % 种群中个体数目
newPop = zeros(n,e);                   % 为返回 pop 分配内存空间
fit = zeros(n,1);                      % 为返回选择概率分配内存
x=zeros(n,2);
x(:,1) =[n:-1:1]';
[y x(:,2)] = sort(oldPop(:,e));
r = q/(1-(1-q)^n);                         % 分布归一化
fit(x(:,2))=r*(1-q).^(x(:,1)-1);           % 生成选择概率
fit = cumsum(fit);                         % 计算概率函数的累积和
rNums=sort(rand(n,1));                     % 生成 n 个排序的随机数
fitIn=1; newIn=1;                          % 循环初始化
while newIn<=n                             % 生成 n 个新个体
  if(rNums(newIn)<fit(fitIn))
    newPop(newIn,:) = oldPop(fitIn,:);
    newIn = newIn+1;
  else
    fitIn = fitIn + 1;
  end
end
```

5. maxGenTerm.m

【功能】当迭代次数大于最大迭代次数时，终止遗传算法，返回为 1，否则返回为 0。

```
function [done] = maxGenTerm(ops,bPop,endPop)
currentGen = ops(1);
maxGen     = ops(2);
done       = currentGen >= maxGen;
```

6. crtbase.m

【功能】生成染色体基向量。

```
function BaseVec = crtbase(Lind, Base)
[ml LenL] = size(Lind) ;
if nargin < 2
     Base = 2 * ones(LenL,1) ;
end
[mb LenB] = size(Base) ;
```

```
% 检验参数一致性
if ml > 1 | mb > 1
      error( 'Lind or Base is not a vector') ;
elseif (LenL > 1 & LenB > 1 & LenL ~= LenB) | (LenL == 1 & LenB > 1 )
      error( 'Vector dimensions must agree' ) :
elseif LenB == 1 & LenL > 1
      Base = Base * ones(LenL,1) ;

end

BaseVec = [] ;
for i = 1:LenL
      BaseVec = [BaseVec, Base(i)*ones(Lind(i),1)'];
end
```

7. xovmp.m

**【功能】多点交叉。**

```
function NewChrom = xovmp(OldChrom, Px, Npt, Rs);
% 种群大小为 Nind and 染色体长度为 Lind
[Nind,Lind] = size(OldChrom);
if Lind < 2, NewChrom = OldChrom; return; end
if nargin < 4, Rs = 0; end
if nargin < 3, Npt = 0; Rs = 0; end
if nargin < 2, Px = 0.7; Npt = 0; Rs = 0; end
if isnan(Px), Px = 0.7; end
if isnan(Npt), Npt = 0; end
if isnan(Rs), Rs = 0; end
if isempty(Px), Px = 0.7; end
if isempty(Npt), Npt = 0; end
if isempty(Rs), Rs = 0; end

Xops = floor(Nind/2);
DoCross = rand(Xops,1) < Px;
odd = 1:2:Nind-1;
even = 2:2:Nind;
% 计算每个染色体对的有效长度
Mask = ~Rs | (OldChrom(odd, :) ~= OldChrom(even, :));
Mask = cumsum(Mask')';
% 根据有效长度和 Px 来计算每个个体对的交叉位置
xsites(:, 1) = Mask(:, Lind);
if Npt >= 2,
          xsites(:, 1) = ceil(xsites(:, 1) .* rand(Xops, 1));
end
xsites(:,2) = rem(xsites + ceil((Mask(:, Lind)-1) .* rand(Xops, 1)) ...
                                    .* DoCross - 1 , Mask(:, Lind) )+1;
% 用 0-1 表示交叉位置
```

```
Mask = (xsites(:,ones(1,Lind)) < Mask) == ...
                        (xsites(:,2*ones(1,Lind)) < Mask);
if ~Npt,
        shuff = rand(Lind,Xops);
        [ans,shuff] = sort(shuff);
        for i=1:Xops
           OldChrom(odd(i),:)=OldChrom(odd(i),shuff(:,i));
           OldChrom(even(i),:)=OldChrom(even(i),shuff(:,i));
        end
end
%交叉
NewChrom(odd,:) = (OldChrom(odd,:).* Mask) + (OldChrom(even,:).*(~Mask));
NewChrom(even,:) = (OldChrom(odd,:).*(~Mask)) + (OldChrom(even,:).*Mask);

% 如果个体数目为奇数，则最后一个个体不能被交叉
%但是必须包含在新一代中
if rem(Nind,2),
   NewChrom(Nind,:)=OldChrom(Nind,:);
end
if ~Npt,
        [ans,unshuff] = sort(shuff);
        for i=1:Xops
           NewChrom(odd(i),:)=NewChrom(odd(i),unshuff(:,i));
           NewChrom(even(i),:)=NewChrom(even(i),unshuff(:,i));
        end
end
```

8. rep.m

**【功能】复制矩阵。**

```
function MatOut = rep(MatIn,REPN)
% 输入矩阵大小
   [N_D,N_L] = size(MatIn);
% 计算
   Ind_D = rem(0:REPN(1)*N_D-1,N_D) + 1;
   Ind_L = rem(0:REPN(2)*N_L-1,N_L) + 1;
% 生成输出矩阵
   MatOut = MatIn(Ind_D,Ind_L);
```

9. ranking.m

**【功能】种群个体的排序**

```
function FitnV = ranking(ObjV, RFun, SUBPOP);
% 向量大小为 Nind
   [Nind,ans] = size(ObjV);
   if nargin < 2, RFun = []; end
   if nargin > 1, if isnan(RFun), RFun = []; end, end
   if prod(size(RFun)) == 2,
      if RFun(2) == 1, NonLin = 1;
```



```
       elseif RFun(2) == 0, NonLin = 0;
       else error('Parameter for ranking method must be 0 or 1'); end
       RFun = RFun(1);
       if isnan(RFun), RFun = 2; end
    elseif prod(size(RFun)) > 2,
       if prod(size(RFun)) ~= Nind, error('ObjV and RFun disagree'); end
    end
    if nargin < 3, SUBPOP = 1; end
    if nargin > 2,
       if isempty(SUBPOP), SUBPOP = 1;
       elseif isnan(SUBPOP), SUBPOP = 1;
       elseif length(SUBPOP) ~= 1, error('SUBPOP must be a scalar'); end
    end
    if (Nind/SUBPOP) ~= fix(Nind/SUBPOP), error('ObjV and SUBPOP disagree'); end
    Nind = Nind/SUBPOP;
   % 检查等级函数。如果需要使用默认值
    if isempty(RFun),
       % 选择压力为 2 的等级函数
          RFun = 2*[0:Nind-1]'/(Nind-1);
    elseif prod(size(RFun)) == 1
       if NonLin == 1,
          % 非线性等级函数
          if RFun(1) < 1, error('Selective pressure must be greater than 1');
          elseif RFun(1) > Nind-2, error('Selective pressure too big'); end
          Root1 = roots([RFun(1)-Nind [RFun(1)*ones(1,Nind-1)]]);
          RFun = (abs(Root1(1)) * ones(Nind,1)) .^ [(0:Nind-1)'];
          RFun = RFun / sum(RFun) * Nind;
       else
          %线性等级函数，且 SP∈[1,2]
          if (RFun(1) < 1 | RFun(1) > 2),
             error('Selective pressure for linear ranking must be between 1 and 2');
          end
          RFun = 2-RFun + 2*(RFun-1)*[0:Nind-1]'/(Nind-1);
       end
   end;
   FitnV = [];
%循环
for irun = 1:SUBPOP,
   % 复制实际子代的目标值
      ObjVSub = ObjV((irun-1)*Nind+1:irun*Nind);

%排序
      NaNix = isnan(ObjVSub);
      Validix = find(~NaNix);
      [ans,ix] = sort(-ObjVSub(Validix));
   % 建立索引向量
```

```
            ix = [find(NaNix) ; Validix(ix)];
        % 得到 ObjV 的排序结果
            Sorted = ObjVSub(ix);
        % 适应度分配
            i = 1;
            FitnVSub = zeros(Nind,1);
            for j = [find(Sorted(1:Nind-1) ~= Sorted(2:Nind))  Nind]',
                FitnVSub(i:j) = sum(RFun(i:j)) * ones(j-i+1,1) / (j-i+1);
                i =j+1;
            end
        % 最后返回没有排序的向量
            [ans,uix] = sort(ix);
            FitnVSub = FitnVSub(uix);
        % 添加 FitnVSub 到 FitnV 中
            FitnV = [FitnV; FitnVSub];
    End
```

10．recombin.m

**【功能】种群个体的重新组合。**

```
function NewChrom = recombin(REC_F, Chrom, RecOpt, SUBPOP);
% 参数一致性检验
    if nargin < 2, error('Not enough input parameter'); end
    %种群大小为 Nind
    [Nind,Nvar] = size(Chrom);
    if nargin < 4, SUBPOP = 1; end
    if nargin > 3,
        if isempty(SUBPOP), SUBPOP = 1
        elseif isnan(SUBPOP), SUBPOP = 1;
        elseif length(SUBPOP) ~= 1, error('SUBPOP must be a scalar'); end
    end
    if (Nind/SUBPOP) ~= fix(Nind/SUBPOP), error('Chrom and SUBPOP disagree'); end
    Nind = Nind/SUBPOP;    % 计算每个子代的个体数目
    if nargin < 3, RecOpt = 0.7; end
    if nargin > 2,
        if isempty(RecOpt), RecOpt = 0.7;
        elseif isnan(RecOpt), RecOpt = 0.7
        elseif length(RecOpt) ~= 1, error('RecOpt must be a scalar');
        elseif (RecOpt < 0 | RecOpt > 1), error('RecOpt must be a scalar in [0, 1]'); end
    end
% 从子代中选择个体
    NewChrom = [];
    for irun = 1:SUBPOP,
        ChromSub = Chrom((irun-1)*Nind+1:irun*Nind, :);
        NewChromSub = feval(REC_F, ChromSub, RecOpt);
        NewChrom=[NewChrom; NewChromSub];
    End
```

11. reins.m

【功能】在种群中重新插入新的子代。

```
function [Chrom, ObjVCh] = reins(Chrom, SelCh, SUBPOP, InsOpt, ObjVCh, ObjVSel);
% 参数一致性检验
   if nargin < 2, error('Not enough input parameter'); end
   if (nargout == 2 & nargin < 6), error('Input parameter missing: ObjVCh and/or ObjVSel'); end
   [NindP, NvarP] = size(Chrom);
   [NindO, NvarO] = size(SelCh);
   if nargin == 2, SUBPOP = 1; end
   if nargin > 2,
      if isempty(SUBPOP), SUBPOP = 1;
      elseif isnan(SUBPOP), SUBPOP = 1;
      elseif length(SUBPOP) ~= 1, error('SUBPOP must be a scalar'); end
   end
   if (NindP/SUBPOP) ~= fix(NindP/SUBPOP), error('Chrom and SUBPOP disagree'); end
   if (NindO/SUBPOP) ~= fix(NindO/SUBPOP), error('SelCh and SUBPOP disagree'); end
   NIND = NindP/SUBPOP;
   NSEL = NindO/SUBPOP;
   IsObjVCh = 0; IsObjVSel = 0;
   if nargin > 4,
      [mO, nO] = size(ObjVCh);
      if nO ~= 1, error('ObjVCh must be a column vector'); end
      if NindP ~= mO, error('Chrom and ObjVCh disagree'); end
      IsObjVCh = 1;
   end
   if nargin > 5,
      [mO, nO] = size(ObjVSel);
      if nO ~= 1, error('ObjVSel must be a column vector'); end
      if NindO ~= mO, error('SelCh and ObjVSel disagree'); end
      IsObjVSel = 1;
   end
   if nargin < 4, INSR = 1.0; Select = 0; end
   if nargin >= 4,
      if isempty(InsOpt), INSR = 1.0; Select = 0;
      elseif isnan(InsOpt), INSR = 1.0; Select = 0;
      else
         INSR = NaN; Select = NaN;
         if (length(InsOpt) > 2), error('Parameter InsOpt too long'); end
         if (length(InsOpt) >= 1), Select = InsOpt(1); end
         if (length(InsOpt) >= 2), INSR = InsOpt(2); end
         if isnan(Select), Select = 0; end
         if isnan(INSR), INSR =1.0; end
      end
   end
   if (INSR < 0 | INSR > 1), error('Parameter for insertion rate must be a scalar in [0, 1]'); end
   if (INSR < 1 & IsObjVSel ~= 1), error('For selection of offspring ObjVSel is needed'); end
```

```
    if (Select ~= 0 & Select ~= 1), error('Parameter for selection method must be 0 or 1'); end
    if (Select == 1 & IsObjVCh == 0), error('ObjVCh for fitness-based exchange needed'); end
    if INSR == 0, return; end
    NIns = min(max(floor(INSR*NSEL+.5),1),NIND);
% 对每个子代进行插入操作
    for irun = 1:SUBPOP,
      % 计算子代个体插入处在父代中的位置
        if Select == 1,
            [Dummy, ChIx] = sort(-ObjVCh((irun-1)*NIND+1:irun*NIND));
        else
            [Dummy, ChIx] = sort(rand(NIND,1));
        end
        PopIx = ChIx((1:NIns)')+ (irun-1)*NIND;
      % 计算 Nins-%个最好子代个体的位置
        if (NIns < NSEL),   % 选择最好的子代个体
            [Dummy,OffIx] = sort(ObjVSel((irun-1)*NSEL+1:irun*NSEL));
        else
            OffIx = (1:NIns)';
        end
        SelIx = OffIx((1:NIns)')+(irun-1)*NSEL;
      % 在子种群中插入子代个体得到新的种群
        Chrom(PopIx,:) = SelCh(SelIx,:);
        if (IsObjVCh == 1 & IsObjVSel == 1), ObjVCh(PopIx) = ObjVSel(SelIx); end
    end
```

12. migrate.m

**【功能】种群个体的迁移。**

```
  function [Chrom, ObjV] = migrate(Chrom, SUBPOP, MigOpt, ObjV);
  % 参数一致性检验
    if nargin < 2, error('Input parameter SUBPOP missing'); end
    if (nargout == 2 & nargin < 4), error('Input parameter ObjV missing'); end
    [Nind, Nvar] = size(Chrom);
    if length(SUBPOP) ~= 1, error('SUBPOP must be a scalar'); end
    if SUBPOP == 1, return; end
    if (Nind/SUBPOP) ~= fix(Nind/SUBPOP), error('Chrom and SUBPOP disagree'); end
    NIND = Nind/SUBPOP;
    if nargin > 3,
        [mO, nO] = size(ObjV);
        if nO ~= 1, error('ObjV must be a column vector'); end
        if Nind ~= mO, error('Chrom and ObjV disagree'); end
        IsObjV = 1;
    else IsObjV = 0; ObjV = [];
    end
    if nargin < 3, MIGR = 0.2; Select = 0; Structure = 0; end
    if nargin > 2,
        if isempty(MigOpt), MIGR = 0.2; Select = 0; Structure = 0;
        elseif isnan(MigOpt), MIGR = 0.2; Select = 0; Structure = 0;
```

```
        else
            MIGR = NaN; Select = NaN; Structure = NaN;
            if length(MigOpt) > 3, error('Parameter MigOpt is too long'); end
            if length(MigOpt) >= 1, MIGR = MigOpt(1); end
            if length(MigOpt) >= 2, Select = MigOpt(2); end
            if length(MigOpt) >= 3, Structure = MigOpt(3); end
            if isnan(MIGR), MIGR =0.2; end
            if isnan(Select), Select = 0; end
            if isnan(Structure), Structure = 0; end
        end
    end

    if (MIGR < 0 | MIGR > 1), error('Parameter for migration rate must be a scalar in [0 1]'); end
    if (Select ~= 0 & Select ~= 1), error('Parameter for selection method must be 0 or 1'); end
    if (Structure < 0 | Structure > 2), error ('Parameter for structure must be 0, 1 or 2'); end
    if (Select == 1 & IsObjV == 0), error('ObjV for fitness-based migration needed');end

    if MIGR == 0, return; end
    MigTeil = max(floor(NIND * MIGR), 1);       % 迁移的个体数目

%子代之间进行迁移
% 在每个子种群中根据最好的个体生成迁移矩阵
    % 清除存储矩阵
        ChromMigAll = [];
        if IsObjV == 1, ObjVAll = []; end
    % 由所有子种群的最优个体生成矩阵
        for irun = 1:SUBPOP
            % 排序 ObjV
              if Select == 1,
                  [Dummy, IndMigSo]=sort(ObjV((irun-1)*NIND+1:irun*NIND));
              else
                  [Dummy, IndMigSo]=sort(rand(NIND, 1));
              end
            % 取出 MigTeil 个最优个体, 复制这些个体及其对应的目标函数值
              IndMigTeil=IndMigSo(1:MigTeil)+(irun-1)*NIND;
              ChromMigAll = [ChromMigAll; Chrom(IndMigTeil,:)];
              if IsObjV == 1, ObjVAll = [ObjVAll; ObjV(IndMigTeil,:)]; end
        end
    % 迁移
        for irun = 1:SUBPOP
          ChromMig = ChromMigAll;
          if IsObjV == 1, ObjVMig = ObjVAll; end
          if Structure == 1,
          % 选择邻近子代的个体得到 ChromMig 和 ObjVMig
              popnum = [SUBPOP 1:SUBPOP 1];
              ins1 = popnum(irun); ins2 = popnum(irun + 2);
```

```
        InsRows = [(ins1-1)*MigTeil+1:ins1*MigTeil (ins2-1)*MigTeil+1:ins2*MigTeil];
        ChromMig = ChromMig(InsRows,:);
        if IsObjV == 1, ObjVMig = ObjVMig(InsRows,:); end
        elseif Structure == 2,
          % 选择当代子代的个体得到 ChromMig 和 ObjVMig
          popnum = [SUBPOP 1:SUBPOP 1];
          ins1 = popnum(irun);
          InsRows = (ins1-1)*MigTeil+1:ins1*MigTeil;
          ChromMig = ChromMig(InsRows,:);
          if IsObjV == 1, ObjVMig = ObjVMig(InsRows,:); end
        else
          % 从 ChromMig 和 ObjVMig 中删除当代子代的个体
          DelRows = (irun-1)*MigTeil+1:irun*MigTeil;
          ChromMig(DelRows,:) = [];
          if IsObjV == 1, ObjVMig(DelRows,:) = []; end
        end
      % 根据任意数目的排序向量生成索引
        [Dummy,IndMigRa]=sort(rand(size(ChromMig,1),1));
      % 从任意向量中取出 MigTeil 个数
        IndMigN=IndMigRa((1:MigTeil)');
      % 复制 MigTeil 的个体到 Chrom 和 ObjV
        Chrom((1:MigTeil)+(irun-1)*NIND,:) = ChromMig(IndMigN,:);
        if IsObjV == 1, ObjV((1:MigTeil)+(irun-1)*NIND,:) = ObjVMig(IndMigN,:); end
end
```

13. crtrp.m

【功能】生成实值的初始种群，crtbp.m 生成二进制的初始种群。

```
function Chrom = crtrp(Nind,FieldDR);
% 参数一致性检验
   if nargin < 2, error('parameter FieldDR missing'); end
   if nargin > 2, nargin = 2; end
   [mN, nN] = size(Nind);
   [mF, Nvar] = size(FieldDR);
   if (mN ~= 1 & nN ~= 1), error('Nind has to be a scalar'); end
   if mF ~= 2, error('FieldDR must be a matrix with 2 rows'); end

   Range = rep((FieldDR(2,:)-FieldDR(1,:)),[Nind 1]);
   Lower = rep(FieldDR(1,:), [Nind 1]);
% 生成初始种群
% 每行包含一个个体，每个变量的值均匀分布在上下边界区间内
   Chrom = rand(Nind,Nvar) .* Range + Lower;
```